Connect. Accelerate. Outperform.™

Mellanox OFED for Linux User Manual

Rev 2.2-1.0.1

Last Updated: July 14, 2014

www.mellanox.com

NOTE:
THIS HARDWARE, SOFTWARE OR TEST SUITE PRODUCT ("PRODUCT(S)") AND ITS RELATED DOCUMENTATION ARE PROVIDED BY MELLANOX TECHNOLOGIES "AS-IS" WITH ALL FAULTS OF ANY KIND AND SOLELY FOR THE PURPOSE OF AIDING THE CUSTOMER IN TESTING APPLICATIONS THAT USE THE PRODUCTS IN DESIGNATED SOLUTIONS. THE CUSTOMER'S MANUFACTURING TEST ENVIRONMENT HAS NOT MET THE STANDARDS SET BY MELLANOX TECHNOLOGIES TO FULLY QUALIFY THE PRODUCTO(S) AND/OR THE SYSTEM USING IT. THEREFORE, MELLANOX TECHNOLOGIES CANNOT AND DOES NOT GUARANTEE OR WARRANT THAT THE PRODUCTS WILL OPERATE WITH THE HIGHEST QUALITY. ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT ARE DISCLAIMED. IN NO EVENT SHALL MELLANOX BE LIABLE TO CUSTOMER OR ANY THIRD PARTIES FOR ANY DIRECT, INDIRECT, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES OF ANY KIND (INCLUDING, BUT NOT LIMITED TO, PAYMENT FOR PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY FROM THE USE OF THE PRODUCT(S) AND RELATED DOCUMENTATION EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Mellanox Technologies
350 Oakmead Parkway Suite 100
Sunnyvale, CA 94085
U.S.A.
www.mellanox.com
Tel: (408) 970-3400
Fax: (408) 970-3403

Mellanox Technologies, Ltd.
Beit Mellanox
PO Box 586 Yokneam 20692
Israel
www.mellanox.com
Tel: +972 (0)74 723 7200
Fax: +972 (0)4 959 3245

© Copyright 2014. Mellanox Technologies. All Rights Reserved.

Mellanox®, Mellanox logo, BridgeX®, ConnectX®, Connect-IB®, CoolBox®, CORE-Direct®, InfiniBridge®, InfiniHost®, InfiniScale®, MetroX®, MLNX-OS®, PhyX®, ScalableHPC®, SwitchX®, UFM®, Virtual Protocol Interconnect® and Voltaire® are registered trademarks of Mellanox Technologies, Ltd.

ExtendX™, FabricIT™, Mellanox Open Ethernet™, Mellanox Virtual Modular Switch™, MetroDX™, TestX™, Unbreakable-Link™ are trademarks of Mellanox Technologies, Ltd.

All other trademarks are property of their respective owners.

Table of Contents

Table of Contents 3
List of Figures 10
List of Tables 11
Chapter 1 Mellanox OFED Overview 21
1.1 Introduction to Mellanox OFED 21
1.2 Mellanox OFED Package 21
1.2.1 ISO Image 21
1.2.2 Software Components 21
1.2.3 Firmware 22
1.2.4 Directory Structure 22
1.3 Architecture 23
1.3.1 mlx4 VPI Driver 23
1.3.2 mlx5 Driver 24
1.3.3 Mid-layer Core 25
1.3.4 ULPs 25
1.3.5 MPI 26
1.3.6 InfiniBand Subnet Manager 26
1.3.7 Diagnostic Utilities 26
1.3.8 Mellanox Firmware Tools 26
1.4 Quality of Service 27
1.5 RDMA over Converged Ethernet (RoCE) 27
1.5.1 Configuring DAPL over RoCE 28
1.5.2 A Detailed Example 29
Chapter 2 Installation 34
2.1 Hardware and Software Requirements 34
2.2 Downloading Mellanox OFED 34
2.3 Installing Mellanox OFED 35
2.3.1 Pre-installation Notes 35
2.3.2 Installation Script 36
2.3.3 Installation Procedure 39
2.3.4 Installation Results 44
2.3.5 Post-installation Notes 45
2.3.6 Installation Logging 45
2.3.7 openibd Script 45
2.4 Updating Firmware After Installation 46
2.5 Installing MLNX_OFED using YUM 47
2.5.1 Setting up MLNX_OFED YUM Repository 47
2.5.2 Installing MLNX_OFED using the YUM Tool 48
2.5.3 Updating Firmware After Installation 49
2.6 Uninstalling Mellanox OFED 49
2.7 Uninstalling Mellanox OFED using the YUM Tool 49

2.8 UEFI Secure Boot 50
2.8.1 Enrolling Mellanox's x.509 Public Key On your Systems 50
Chapter 3 Configuration Files 51
3.1 Persistent Naming for Network Interfaces. 51
Chapter 4 Driver Features 52
4.1 SCSI RDMA Protocol 52
4.1.1 Overview 52
4.1.2 SRP Initiator 52
4.2 iSCSI Extensions for RDMA (iSER). 61
4.2.1 Overview 61
4.2.2 iSER Initiator 61
4.3 Ethernet over IB (EoIB) vNic 61
4.3.1 Ethernet over IB Topology 62
4.3.2 EoIB Configuration 63
4.3.3 Retrieving EoIB Information 68
4.3.4 Advanced EoIB Settings 72
4.4 IP over InfiniBand 76
4.4.1 Introduction 76
4.4.2 IPoIB Mode Setting. 76
4.4.3 IPoIB Configuration 77
4.4.4 Subinterfaces 80
4.4.5 Verifying IPoIB Functionality 81
4.4.6 Bonding IPoIB 82
4.5 Quality of Service InfiniBand 83
4.5.1 Quality of Service Overview. 83
4.5.2 QoS Architecture. 84
4.5.3 Supported Policy 84
4.5.4 CMA Features 85
4.5.5 OpenSM Features 86
4.6 Quality of Service Ethernet 86
4.6.1 Quality of Service Overview. 86
4.6.2 Mapping Traffic to Traffic Classes 86
4.6.3 Plain Ethernet Quality of Service Mapping 86
4.6.4 RoCE Quality of Service Mapping 87
4.6.5 Raw Ethernet QP Quality of Service Mapping 88
4.6.6 Map Priorities with tc_wrap.py/mlnx_qos 88
4.6.7 Quality of Service Properties 89
4.6.8 Quality of Service Tools 89
4.7 Ethernet Time-Stamping 94
4.7.1 Ethernet Time-Stamping Service 94
4.7.2 RoCE Time Stamping 97
4.8 Ethernet VXLAN. 99
4.8.1 Prerequisites 99
4.8.2 Enabling VXLAN 99
4.8.3 Important Notes. 100

4.9 Atomic Operations . . . 100
4.9.1 Atomic Operations in mlx5 Driver . . . 100
4.9.2 Enhanced Atomic Operations . . . 100
4.10 Ethernet Tunneling Over IPoIB Driver (eIPoIB) . . . 101
4.10.1 Enabling the eIPoIB Driver . . . 102
4.10.2 Configuring the Ethernet Tunneling Over IPoIB Driver . . . 102
4.10.3 VLAN Configuration Over an eIPoIB Interface . . . 104
4.10.4 Setting Performance Tuning . . . 105
4.11 Contiguous Pages . . . 105
4.12 Shared Memory Region . . . 106
4.13 XRC - eXtended Reliable Connected Transport Service for InfiniBand . . . 107
4.14 Flow Steering . . . 107
4.14.1 Enable/Disable Flow Steering . . . 108
4.14.2 Flow Domains and Priorities . . . 108
4.15 Single Root IO Virtualization (SR-IOV) . . . 111
4.15.1 System Requirements . . . 111
4.15.2 Setting Up SR-IOV . . . 112
4.15.3 Enabling SR-IOV and Para Virtualization on the Same Setup . . . 118
4.15.4 Assigning a Virtual Function to a Virtual Machine . . . 119
4.15.5 Uninstalling SR-IOV Driver . . . 120
4.15.6 Configuring Pkeys and GUIDs under SR-IOV . . . 120
4.15.7 Running Network Diagnostic Tools on a Virtual Function . . . 128
4.16 Quantized Congestion Control . . . 129
4.16.1 QCN Tool - mlnx_qcn . . . 130
4.16.2 Setting QCN Configuration . . . 132
4.17 CORE-Direct . . . 132
4.17.1 CORE-Direct Overview . . . 132
4.18 Ethtool . . . 132
4.19 Dynamically Connected Transport Service . . . 134
4.20 PeerDirect . . . 135
4.21 Inline-Receive . . . 135
4.21.1 Querying Inline-Receive Capability . . . 135
4.21.2 Activating Inline-Receive . . . 135
4.22 Ethernet Performance Counters . . . 136
4.23 Memory Window . . . 140
4.23.1 Query Capabilities . . . 141
4.23.2 Allocating Memory Window . . . 141
4.23.3 Binding Memory Windows . . . 141
4.23.4 Invalidating Memory Window . . . 141
4.23.5 Deallocating Memory Window . . . 141
4.24 pm_qos usage on ingress Packet Traffic . . . 142
4.25 XOR RSS Hash Function . . . 142
Chapter 5 HPC Features . . . 143
5.1 HPC-X . . . 143
5.2 Shared Memory Access . . . 143

5.2.1 Mellanox ScalableSHMEM 144
5.2.2 Running SHMEM with FCA 144
5.2.3 Running ScalableSHMEM with MXM 145
5.2.4 Running SHMEM with Contiguous Pages 145
5.2.5 Running ScalableSHMEM Application 146
5.3 Message Passing Interface 146
5.3.1 Overview 146
5.3.2 Prerequisites for Running MPI 146
5.3.3 MPI Selector - Which MPI Runs 148
5.3.4 Compiling MPI Applications 148
5.4 MellanoX Messaging 148
5.4.1 Compiling Open MPI with MXM 149
5.4.2 Enabling MXM in OpenMPI 150
5.4.3 Tuning MXM Settings 150
5.4.4 Configuring Multi-Rail Support 150
5.4.5 Configuring MXM over the Ethernet Fabric 151
5.4.6 Configuring MXM over Different Transports 151
5.4.7 Configuring Service Level Support 152
5.5 Fabric Collective Accelerator 152
5.6 ScalableUPC 154
5.6.1 Installing ScalableUPC 155
5.6.2 FCA Runtime Parameters 155
5.6.3 Various Executable Examples 156
Chapter 6 Working With VPI 158
6.1 Port Type Management 158
6.2 Auto Sensing 158
6.2.1 Enabling Auto Sensing 158
Chapter 7 Performance 160
Chapter 8 OpenSM – Subnet Manager 161
8.1 Overview 161
8.2 opensm Description 161
8.2.1 opensm Syntax 161
8.2.2 Environment Variables 169
8.2.3 Signaling 170
8.2.4 Running opensm 170
8.3 osmtest Description 170
8.3.1 Syntax 171
8.3.2 Running osmtest 173
8.4 Partitions 173
8.4.1 File Format 173
8.5 Routing Algorithms 176
8.5.1 Effect of Topology Changes 177
8.5.2 Min Hop Algorithm 177
8.5.3 UPDN Algorithm 178
8.5.4 Fat-tree Routing Algorithm 179

8.5.5 LASH Routing Algorithm . . . 180
8.5.6 DOR Routing Algorithm . . . 182
8.5.7 Torus-2QoS Routing Algorithm . . . 182
8.5.8 Routing Chains . . . 190
8.6 Quality of Service Management in OpenSM . . . 198
8.6.1 Overview . . . 198
8.6.2 Advanced QoS Policy File . . . 199
8.6.3 Simple QoS Policy Definition . . . 200
8.6.4 Policy File Syntax Guidelines . . . 200
8.6.5 Examples of Advanced Policy File . . . 201
8.6.6 Simple QoS Policy - Details and Examples . . . 203
8.6.7 SL2VL Mapping and VL Arbitration . . . 206
8.6.8 Deployment Example . . . 207
8.7 QoS Configuration Examples . . . 208
8.7.1 Typical HPC Example: MPI and Lustre . . . 208
8.7.2 EDC SOA (2-tier): IPoIB and SRP . . . 209
8.7.3 EDC (3-tier): IPoIB, RDS, SRP . . . 210
8.8 Adaptive Routing . . . 211
8.8.1 Overview . . . 211
8.8.2 Installing the Adaptive Routing . . . 211
8.8.3 Running Subnet Manager with Adaptive Routing Manager . . . 211
8.8.4 Querying Adaptive Routing Tables . . . 212
8.8.5 Adaptive Routing Manager Options File . . . 213
8.9 Congestion Control . . . 216
8.9.1 Congestion Control Overview . . . 216
8.9.2 Running OpenSM with Congestion Control Manager . . . 216
8.9.3 Configuring Congestion Control Manager . . . 216
8.9.4 Configuring Congestion Control Manager Main Settings . . . 217
Chapter 9 InfiniBand Fabric Utilities . . . 220
9.1 Common Configuration, Interface and Addressing . . . 220
9.2 InfiniBand Interface Definition . . . 220
9.3 Addressing . . . 220
9.4 Diagnostic Utilities . . . 221
9.4.1 ibdiagnet (of ibutils2) - IB Net Diagnostic . . . 221
9.4.2 ibdiagnet (of ibutils) - IB Net Diagnostic . . . 226
9.4.3 ibdiagpath - IB Diagnostic Path . . . 226
9.4.4 ibstat . . . 228
9.4.5 ibtracert . . . 228
9.4.6 ibqueryerrors . . . 230
9.4.7 iblinkinfo . . . 232
9.4.8 saquery . . . 233
9.4.9 smpdump . . . 235
9.4.10 ibv_devices . . . 237
9.4.11 ibv_devinfo . . . 237
9.4.12 ibdev2netdev . . . 238

9.4.13 ibstatus . . . 239
9.4.14 ibportstate . . . 241
9.4.15 ibroute . . . 244
9.4.16 smpquery . . . 247
9.4.17 perfquery . . . 250
9.4.18 mstflint . . . 253
9.4.19 ibv_asyncwatch . . . 258
9.4.20 ibdump . . . 258
9.4.21 ibnodes . . . 259
9.4.22 ibswitches . . . 260
9.4.23 ibnetsplit . . . 260
9.4.24 ibccconfig . . . 261
9.4.25 ibccquery . . . 262
9.4.26 smparquery . . . 263
9.4.27 ibtopodiff . . . 264
9.4.28 ibcongest . . . 265
9.4.29 ibaddr . . . 267
9.4.30 sminfo . . . 267
9.4.31 ibnetdiscover . . . 268
9.4.32 ibping . . . 268
9.4.33 ibsysstat . . . 269
9.4.34 ibcacheedit . . . 269
9.4.35 dump_fts . . . 270
9.5 Performance Utilities . . . 272
9.5.1 ib_read_bw . . . 272
9.5.2 ib_read_lat . . . 274
9.5.3 ib_send_bw . . . 275
9.5.4 ib_send_lat . . . 278
9.5.5 ib_write_bw . . . 280
9.5.6 ib_write_lat . . . 282
9.5.7 ib_atomic_bw . . . 283
9.5.8 ib_atomic_lat . . . 285
9.5.9 raw_ethernet_bw . . . 287
9.5.10 raw_ethernet_lat . . . 291
Appendix A SRP Target Driver . . . 294
A.1 Prerequisites and Installation . . . 294
A.2 How-to Run . . . 294
A.3 How-to Unload/Shutdown . . . 297
Appendix B mlx4 Module Parameters . . . 298
B.1 mlx4_ib Parameters . . . 298
B.2 mlx4_core Parameters . . . 298
B.3 mlx4_en Parameters . . . 299
Appendix C mlx5 Module Parameters . . . 300
Appendix D Lustre Compilation over MLNX_OFED . . . 301
Appendix E Using FlexBoot for Booting Various OSs from an iSCSI Target 302

E.1 Configuring the iSCSI Target Machine 302
E.2 Configuring the DHCP Server 304
E.3 Configuring the PXE Server 304
E.4 Installing SLES11 SP3 on a Remote Storage over iSCSI 305
E.5 Using PXE Boot Services for Booting the SLES11 SP3 from the iSCSI Target 312
E.6 Installing RHEL6.4 on a Remote Storage over iSCSI 312
E.7 SAN-Booting the Diskless Client with FlexBoot 316
E.8 Sanity Checks 316
E.9 iSCSI Login 316
E.10 DHCP Configuration for PXELINUX with FlexBoot 318
E.11 pxelinux.cfg/default 319
E.12 DHCP Configuration for iSCSI Boot with FlexBoot (PXE SAN Boot) 319
E.13 SAN-Booting with FlexBoot in CHAP Environment 319

List of Figures

Figure 1: Mellanox OFED Stack for ConnectX® Family Adapter Cards 23
Figure 2: I/O Consolidation Over InfiniBand 83
Figure 3: An Example of a Virtual Network 104
Figure 4: FCA Architecture 153
Figure 5: FCA Components 154
Figure 6: QoS Manager 199
Figure 7: Example QoS Deployment on InfiniBand Subnet 208

List of Tables

Table 1: Document Revision History 13
Table 2: Abbreviations and Acronyms 17
Table 3: Glossary 18
Table 4: Reference Documents 20
Table 5: Software and Hardware Requirements 34
Table 6: mlnxofedinstall Return Codes 38
Table 7: mlx4_vnic.conf file format 63
Table 8: Red Hat Linux mlx4_vnic.conf file format 64
Table 9: Buffer Values 105
Table 10: Parameters Used to Control Error Cases / Contiguity 106
Table 11: Flow Specific Parameters 110
Table 12: ethtool Supported Options 133
Table 13: Port IN Counters 136
Table 14: Port OUT Counters 137
Table 15: Port VLAN Priority Tagging (where <i> is in the range 0…7) 138
Table 16: Port Pause (where <i> is in the range 0…7) 138
Table 17: VPort Statistics (where <i>=<empty_string> is the PF, and ranges 1…NumOfVf per VF) 139
Table 18: SW Statistics 140
Table 19: Per Ring (SW) Statistics (where <i> is the ring I – per configuration) 140
Table 20: Useful MPI Links 146
Table 21: MLNX_OFED and MXM Versions 149
Table 22: Runtime Parameters 155
Table 23: Port Group Qualifiers 191
Table 24: Port Group Qualifiers 192
Table 25: Topology Qualifiers 195
Table 26: Routing Engine Qualifiers 196
Table 27: Adaptive Routing Manager Options File 214
Table 28: Adaptive Routing Manager Pre-Switch Options File 215
Table 29: Congestion Control Manager General Options File 218
Table 30: Congestion Control Manager Switch Options File 218
Table 31: Congestion Control Manager CA Options File 218
Table 32: Congestion Control Manager CC MGR Options File 219
Table 33: ibdiagnet (of ibutils2) Output Files 223
Table 34: ibdiagpath Output Files 227
Table 35: ibstat Flags and Options 228

Table 36: ibtracert Flags and Options 229
Table 37: ibqueryerrors Flags and Options 230
Table 38: iblinkinfo Flags and Options 232
Table 39: saquery Flags and Options 233
Table 40: smpdump Flags and Options 235
Table 41: ibv_devinfo Flags and Options 237
Table 42: ibstatus Flags and Options 239
Table 43: ibportstate Flags and Options 244
Table 44: smpquery Flags and Options 247
Table 45: perfquery Options 250
Table 46: mstflint Switches 254
Table 47: mstflint Commands 255
Table 48: ib_read_bw Flags and Options 272
Table 49: Additional ib_read_bw Flags and Options 273
Table 50: ib_read_lat Flags and Options 274
Table 51: Additional ib_read_lat Flags and Options 275
Table 52: ib_send_bw Flags and Options 276
Table 53: Additional ib_send_bw Flags and Options 277
Table 54: Additional Rate Limiter Flags and Options 278
Table 55: ib_send_lat Flags and Options 278
Table 56: Additional ib_send_lat Flags and Options 279
Table 57: ib_write_bw Flags and Options 280
Table 58: Additional ib_write_bw Flags and Options 281
Table 59: ib_write_lat Flags and Options 282
Table 60: Additional ib_write_lat Flags and Options 283
Table 61: ib_atomic_bw Flags and Options 284
Table 62: Additional ib_atomic_bw Flags and Options 285
Table 63: ib_atomic_lat Flags and Options 286
Table 64: Additional ib_atomic_lat Flags and Options 287
Table 65: raw_ethernet_bw Flags and Options 287
Table 66: Additional raw_ethernet_bw Flags and Options 289
Table 67: raw_ethernet_bw Rate Limiter Flags and Options 290
Table 68: raw_ethernet_bw Raw Ethernet Flags and Options 290
Table 69: raw_ethernet_lat Flags and Options 291
Table 70: Additional raw_ethernet_lat Flags and Options 292
Table 71: raw_ethernet_lat Raw Ethernet Flags and Options 292

Document Revision History

Table 1 - Document Revision History

Release	Date	Description
2.2-1.0.1	June 10, 2014	• Added the following sections: • Section 1.5.1, "Configuring DAPL over RoCE", on page 28 • Section 1.5.2, "A Detailed Example", on page 29 and its subsections • Section 2.8, "UEFI Secure Boot", on page 50 • Section 4.3, "Ethernet over IB (EoIB) vNic", on page 62 and its subsections • Section 5.4.6, "Configuring MXM over Different Transports", on page 151 • Section 5.4.7, "Configuring Service Level Support", on page 152 • Section 8.5.8, "Routing Chains", on page 190 and its subsections • Section 9.4.21, "ibnodes", on page 259 • Section 9.4.22, "ibswitches", on page 260 • Section 9.4.23, "ibnetsplit", on page 260 • Section 9.4.24, "ibccconfig", on page 261 • Section 9.4.25, "ibccquery", on page 262 • Section 9.4.26, "smparquery", on page 263 • Section 9.4.27, "ibtopodiff", on page 264 • Section 9.4.28, "ibcongest", on page 265 • Section 9.4.29, "ibaddr", on page 267 • Section 9.4.30, "sminfo", on page 267 • Section 9.4.31, "ibnetdiscover", on page 268 • Section 9.4.32, "ibping", on page 268 • Section 9.4.33, "ibsysstat", on page 269 • Section 9.4.34, "ibcacheedit", on page 269 • Section 9.4.35, "dump_fts", on page 270 • Updated thew following sections: • Section 5.5, "Fabric Collective Accelerator", on page 152 • Section 8.5.1, "Effect of Topology Changes", on page 177 • Section 8.5.3.1, "UPDN Algorithm Usage", on page 179 • Section 9.4.1, "ibdiagnet (of ibutils2) - IB Net Diagnostic", on page 221 • Section 9.4.2, "ibdiagnet (of ibutils) - IB Net Diagnostic", on page 226 • Section 9.4.6, "ibqueryerrors", on page 230 • Table 33, "ibqueryerrors Flags and Options," on page 230 • Table 41, "perfquery Options," on page 250

Table 1 - Document Revision History

Release	Date	Description
2.2-1.0.1	April 30, 2014	• Added the following sections: • Section 2.3.7, “openibd Script”, on page 45 • Section 4.8, “Ethernet VXLAN”, on page 99 • Section 4.9.1, “Atomic Operations in mlx5 Driver”, on page 100 • Section 4.15.7, “Running Network Diagnostic Tools on a Virtual Function”, on page 128 • Section 4.15.7.2, “Granting SMP Capability to a Virtual Function”, on page 129 • Section 4.15.7.3, “Installing MLNX_OFED with Network Diagnostics on a VM”, on page 129 • Section 4.16, “Quantized Congestion Control”, on page 129 • Section 4.16.1, “QCN Tool - mlnx_qcn”, on page 130 • Section 4.16.2, “Setting QCN Configuration”, on page 132 • Section 4.15.6.2.3, “Multi-GUID Support in InfiniBand”, on page 123 • Section 4.15.6.3.4, “Mapping VFs to Ports using the mlnx_get_vfs.pl tool”, on page 128 • Section 4.24, “pm_qos usage on ingress Packet Traffic”, on page 142 • Section 4.25, “XOR RSS Hash Function”, on page 142 • Section 9.4.4, “ibstat”, on page 228 • Section 9.4.5, “ibtracert”, on page 228 • Section 9.4.6, “ibqueryerrors”, on page 230 • Section 9.4.7, “iblinkinfo”, on page 232 • Section 9.4.8, “saquery”, on page 233 • Section 9.4.9, “smpdump”, on page 235 • Section 9.4.10, “ibv_devices”, on page 237 • Section 9.4.10, “ibv_devices”, on page 237 • Section 9.4.21, “ibnodes”, on page 259 • Section 9.4.22, “ibswitches”, on page 260 • Section 9.4.23, “ibnetsplit”, on page 260 • Section 9.5.1, “ib_read_bw”, on page 272 • Section 9.5.2, “ib_read_lat”, on page 274 • Section 9.5.3, “ib_send_bw”, on page 275 • Section 9.5.4, “ib_send_lat”, on page 278 • Section 9.5.5, “ib_write_bw”, on page 280 • Section 9.5.6, “ib_write_lat”, on page 282 • Section 9.5.7, “ib_atomic_bw”, on page 283 • Section 9.5.8, “ib_atomic_lat”, on page 285 • Section 9.5.9, “raw_ethernet_bw”, on page 287 • Section 9.5.10, “raw_ethernet_lat”, on page 291

Table 1 - Document Revision History

Release	Date	Description
2.2-1.0.1	April 30, 2014	• Updated the following section: • Section 2.3.1, "Pre-installation Notes", on page 35 • Section 2.3.2, "Installation Script", on page 36 • Section , "Options", on page 36 • Section 2.3.3, "Installation Procedure", on page 39 • Section 2.3.6, "Installation Logging", on page 45 • Section 2.5.1, "Setting up MLNX_OFED YUM Repository", on page 47 • Section 4.6.4, "RoCE Quality of Service Mapping", on page 87 • Section 4.7.2.2, "Creating Time Stamping Completion Queue", on page 98 • Section 4.7.2.3, "Polling a Completion Queue", on page 98 • Section 4.7.2.4, "Querying the Hardware Time", on page 99 • Section 4.11, "Contiguous Pages", on page 105 • Section 4.15.2, "Setting Up SR-IOV", on page 112 • Section 4.15.6.2, "Virtual Function InfiniBand Ports", on page 122 • Section 4.18, "Ethtool", on page 132 • Appendix D, "Lustre Compilation over MLNX_OFED" page 301
2.2-1.0.1	April 30, 2014	• Removed the following section: • ibcheckerrs • FlexBoot Package • Mellanox FlexBoot • Burning Firmware with SR-IOV
2.1-1.0.6	March 04, 2014	• No changes
2.1-1.0.0	February 18, 2014	• Updated the following section: • Section 2.3.3, "Installation Procedure", on page 39 • Section 4.15.2, "Setting Up SR-IOV", on page 112 • Section 8.6.1, "Overview", on page 198
	December 2013	• Added the following sections: • Section 2.3.6, "Installation Logging", on page 45 • Section 4.7.2, "RoCE Time Stamping", on page 97 and its subsections • Section 4.20, "PeerDirect", on page 135 • Section 4.21, "Inline-Receive", on page 135 • Section 4.22, "Ethernet Performance Counters", on page 136 • Section 4.23, "Memory Window", on page 140 • Section 4.1.2.1.1, "SRP Module Parameters", on page 52 • Section 4.1.2.1.2, "SRP Remote Ports Parameters", on page 53 • Section 4.1.2.2.1, "SRP sysfs Parameters", on page 54 • "srpd" on page 57 • Section 4.7.1.3, "Querying Time Stamping Capabilities via ethtool", on page 97

Table 1 - Document Revision History

Release	Date	Description
2.1-1.0.0	December 2013	• Updated the following sections: • Section 1.5, "RDMA over Converged Ethernet (RoCE)", on page 27 • Section 2.3.3, "Installation Procedure", on page 39 • Section 4.15.2, "Setting Up SR-IOV", on page 112 • Section 5.4.1, "Compiling Open MPI with MXM", on page 149 • Section 5.4.2, "Enabling MXM in OpenMPI", on page 150 • Section 5.4.4, "Configuring Multi-Rail Support", on page 150 • Section 4.10.4, "Setting Performance Tuning", on page 105 • Section 8.4.1, "File Format", on page 173 • Appendix B, "mlx4 Module Parameters" page 298 • Section 4.1.2.2, "Manually Establishing an SRP Connection", on page 54 • Section 4.1.2.3, "SRP Tools - ibsrpdm, srp_daemon and srpd Service Script", on page 56
2.1-1.0.0	December 2013	• Section 4.1.2.4, "Automatic Discovery and Connection to Targets", on page 58 • Section 4.1.2.5, "Multiple Connections from Initiator InfiniBand Port to the Target", on page 59 • Section 4.1.2.6, "High Availability (HA)", on page 59 • Section 4.1.2.7, "Shutting Down SRP", on page 60 • Section 4.18, "Ethtool", on page 132
2.0-3.0.0	October 2013	Removed section "Command Line Interface (CLI)" Updated the following sections: • Appendix , "" page 300
	August 2013	• Updated the following sections: • Section 1.3.4, "ULPs", on page 25 • Section 4.14, "Flow Steering", on page 107 and its subsections • Section 1.3.3, "Mid-layer Core", on page 25 • Section 4.10, "Ethernet Tunneling Over IPoIB Driver (eIPoIB)", on page 101 • Section 8.2.1, "opensm Syntax", on page 161 • Appendix C, "mlx5 Module Parameters" page 300 • Added the following sections: • Section 1.5, "RDMA over Converged Ethernet (RoCE)", on page 27 • Section 4.6, "Quality of Service Ethernet", on page 86 and its subsections • Section 4.13, "XRC - eXtended Reliable Connected Transport Service for InfiniBand", on page 107 • Section 4.15.6, "Configuring Pkeys and GUIDs under SR-IOV", on page 120 and its subsections • Section 4.18, "Ethtool", on page 132
2.0-2.0.5	April 2013	Initial release

About this Manual

This preface provides general information concerning the scope and organization of this User's Manual.

Intended Audience

This manual is intended for system administrators responsible for the installation, configuration, management and maintenance of the software and hardware of VPI (InfiniBand, Ethernet) adapter cards. It is also intended for application developers.

Common Abbreviations and Acronyms

Table 2 - Abbreviations and Acronyms (Sheet 1 of 2)

Abbreviation / Acronym	Whole Word / Description
B	(Capital) 'B' is used to indicate size in bytes or multiples of bytes (e.g., 1KB = 1024 bytes, and 1MB = 1048576 bytes)
b	(Small) 'b' is used to indicate size in bits or multiples of bits (e.g., 1Kb = 1024 bits)
FW	Firmware
HCA	Host Channel Adapter
HW	Hardware
IB	InfiniBand
iSER	iSCSI RDMA Protocol
LSB	Least significant *byte*
lsb	Least significant *bit*
MSB	Most significant *byte*
msb	Most significant *bit*
NIC	Network Interface Card
SW	Software
VPI	Virtual Protocol Interconnect
IPoIB	IP over InfiniBand
PFC	Priority Flow Control
PR	Path Record
RDS	Reliable Datagram Sockets
RoCE	RDMA over Converged Ethernet

Table 2 - Abbreviations and Acronyms (Sheet 2 of 2)

Abbreviation / Acronym	Whole Word / Description
SDP	Sockets Direct Protocol
SL	Service Level
SRP	SCSI RDMA Protocol
MPI	Message Passing Interface
EoIB	Ethernet over Infiniband
QoS	Quality of Service
ULP	Upper Level Protocol
VL	Virtual Lane
vHBA	Virtual SCSI Host Bus adapter
uDAPL	User Direct Access Programming Library

Glossary

The following is a list of concepts and terms related to InfiniBand in general and to Subnet Managers in particular. It is included here for ease of reference, but the main reference remains the *InfiniBand Architecture Specification*.

Table 3 - Glossary (Sheet 1 of 2)

Channel Adapter (CA), Host Channel Adapter (HCA)	An IB device that terminates an IB link and executes transport functions. This may be an HCA (Host CA) or a TCA (Target CA).
HCA Card	A network adapter card based on an InfiniBand channel adapter device.
IB Devices	Integrated circuit implementing InfiniBand compliant communication.
IB Cluster/Fabric/ Subnet	A set of IB devices connected by IB cables.
In-Band	A term assigned to administration activities traversing the IB connectivity only.
Local Identifier (ID)	An address assigned to a port (data sink or source point) by the Subnet Manager, unique within the subnet, used for directing packets within the subnet.
Local Device/Node/ System	The IB Host Channel Adapter (HCA) Card installed on the machine running IBDIAG tools.

Table 3 - Glossary (Sheet 2 of 2)

Local Port	The IB port of the HCA through which IBDIAG tools connect to the IB fabric.
Master Subnet Manager	The Subnet Manager that is authoritative, that has the reference configuration information for the subnet. See Subnet Manager.
Multicast Forwarding Tables	A table that exists in every switch providing the list of ports to forward received multicast packet. The table is organized by MLID.
Network Interface Card (NIC)	A network adapter card that plugs into the PCI Express slot and provides one or more ports to an Ethernet network.
Standby Subnet Manager	A Subnet Manager that is currently quiescent, and not in the role of a Master Subnet Manager, by agency of the master SM. See Subnet Manager.
Subnet Administrator (SA)	An application (normally part of the Subnet Manager) that implements the interface for querying and manipulating subnet management data.
Subnet Manager (SM)	One of several entities involved in the configuration and control of the an IB fabric.
Unicast Linear Forwarding Tables (LFT)	A table that exists in every switch providing the port through which packets should be sent to each LID.
Virtual Protocol Interconnet (VPI)	A Mellanox Technologies technology that allows Mellanox channel adapter devices (ConnectX®) to simultaneously connect to an InfiniBand subnet and a 10GigE subnet (each subnet connects to one of the adpater ports)

Related Documentation

Table 4 - Reference Documents

Document Name	Description
InfiniBand Architecture Specification, Vol. 1, Release 1.2.1	The InfiniBand Architecture Specification that is provided by IBTA
IEEE Std 802.3ae™-2002 (Amendment to IEEE Std 802.3-2002) Document # PDF: SS94996	Part 3: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications Amendment: Media Access Control (MAC) Parameters, Physical Layers, and Management Parameters for 10 Gb/s Operation
Firmware Release Notes for Mellanox adapter devices	See the Release Notes PDF file relevant to your adapter device under `docs/` folder of installed package.
MFT User's Manual	Mellanox Firmware Tools User's Manual. See under `docs/` folder of installed package.
MFT Release Notes	Release Notes for the Mellanox Firmware Tools. See under `docs/` folder of installed package.

Support and Updates Webpage

Please visit http://www.mellanox.com > Products > InfiniBand/VPI Drivers > Linux SW/Drivers for downloads, FAQ, troubleshooting, future updates to this manual, etc.

1 Mellanox OFED Overview

1.1 Introduction to Mellanox OFED

Mellanox OFED is a single Virtual Protocol Interconnect (VPI) software stack which operates across all Mellanox network adapter solutions supporting 10, 20, 40 and 56 Gb/s InfiniBand (IB); 10, 40 and 56 Gb/s Ethernet; and 2.5 or 5.0 GT/s PCI Express 2.0 and 8 GT/s PCI Express 3.0 uplinks to servers.

All Mellanox network adapter cards are compatible with OpenFabrics-based RDMA protocols and software, and are supported with major operating system distributions.

Mellanox OFED is certified with the following products:

- Mellanox Messaging Accelerator (VMA™) software: Socket acceleration library that performs OS bypass for standard socket based applications.
- Mellanox Unified Fabric Manager (UFM®) software: Powerful platform for managing demanding scale-out computing fabric environments, built on top of the OpenSM industry standard routing engine.
- Fabric Collective Accelerator (FCA) - FCA is a Mellanox MPI-integrated software package that utilizes CORE-Direct technology for implementing the MPI collectives communications.

1.2 Mellanox OFED Package

1.2.1 ISO Image

Mellanox OFED for Linux (MLNX_OFED_LINUX) is provided as ISO images or as a tarball, one per supported Linux distribution and CPU architecture, that includes *source code* and *binary* RPMs, firmware, utilities, and documentation. The ISO image contains an installation script (called `mlnxofedinstall`) that performs the necessary steps to accomplish the following:

- Discover the currently installed kernel
- Uninstall any InfiniBand stacks that are part of the standard operating system distribution or another vendor's commercial stack
- Install the MLNX_OFED_LINUX binary RPMs (if they are available for the current kernel)
- Identify the currently installed InfiniBand HCAs and perform the required firmware updates

1.2.2 Software Components

MLNX_OFED_LINUX contains the following software components:

- Mellanox Host Channel Adapter Drivers
  - mlx5, mlx4 (VPI), which is split into multiple modules:
    - mlx4_core (low-level helper)
    - mlx4_ib (IB)
    - mlx5_ib
    - mlx5_core

    - mlx4_en (Ethernet)
- Mid-layer core
    - Verbs, MADs, SA, CM, CMA, uVerbs, uMADs
- Upper Layer Protocols (ULPs)
    - IPoIB, RDS*, SRP Initiator and SRP

      * **NOTE:** RDS was not tested by Mellanox Technologies.
- MPI
    - Open MPI stack supporting the InfiniBand, RoCE and Ethernet interfaces
    - OSU MVAPICH stack supporting the InfiniBand and RoCE interfaces
    - MPI benchmark tests (OSU BW/LAT, Intel MPI Benchmark, Presta)
- OpenSM: InfiniBand Subnet Manager
- Utilities
    - Diagnostic tools
    - Performance tests
- Firmware tools (MFT)
- Source code for all the OFED software modules (for use under the conditions mentioned in the modules' LICENSE files)
- Documentation

1.2.3 Firmware

The ISO image includes the following firmware items:

- Firmware images (.mlx format) for ConnectX®-3/ConnectX®-3 Pro/Connect-IB™ network adapters
- Firmware configuration (.INI) files for Mellanox standard network adapter cards and custom cards
- FlexBoot for ConnectX®-3 HCA devices

1.2.4 Directory Structure

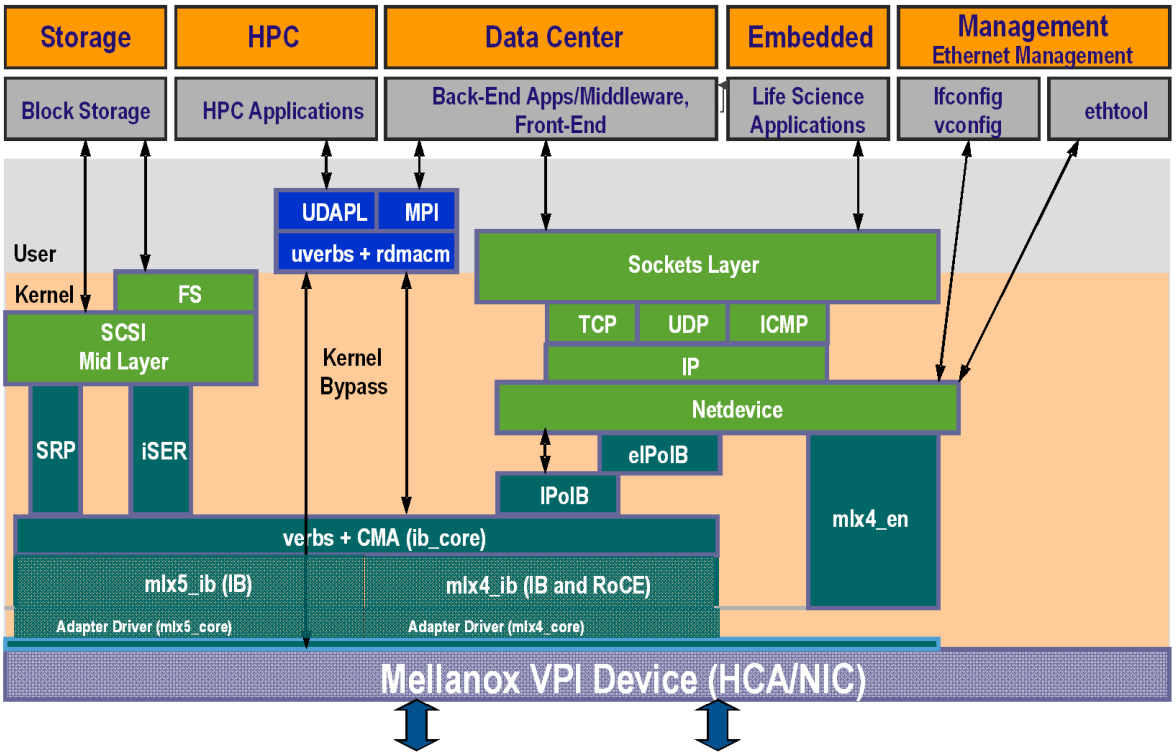
The ISO image of MLNX_OFED_LINUX contains the following files and directories:

- mlnxofedinstall - This is the MLNX_OFED_LINUX installation script.
- ofed_uninstall.sh - This is the MLNX_OFED_LINUX un-installation script.
- <RPMS folders> - Directory of binary RPMs for a specific CPU architecture.
- firmware/ - Directory of the Mellanox IB HCA firmware images (including Boot-over-IB)
- src/ - Directory of the OFED source tarball
- mlnx_add_kernel_support.sh - Script required to rebuild MLNX_OFED_LINUX for customized kernel version on supported Linux Distribution
- docs/ - Directory of Mellanox OFED related documentation

1.3 Architecture

Figure 1 shows a diagram of the Mellanox OFED stack, and how upper layer protocols (ULPs) interface with the hardware and with the kernel and user space. The application level also shows the versatility of markets that Mellanox OFED applies to.

Figure 1: Mellanox OFED Stack for ConnectX® Family Adapter Cards



The following sub-sections briefly describe the various components of the Mellanox OFED stack.

1.3.1 mlx4 VPI Driver

`mlx4` is the low level driver implementation for the ConnectX family adapters designed by Mellanox Technologies. ConnectX® family adapters can operate as an InfiniBand adapter, or as an Ethernet NIC. The OFED driver supports InfiniBand and Ethernet NIC configurations. To accommodate the supported configurations, the driver is split into the following modules:

mlx4_core

Handles low-level functions like device initialization and firmware commands processing. Also controls resource allocation so that the InfiniBand and Ethernet functions can share the device without interfering with each other.

mlx4_ib

Handles InfiniBand-specific functions and plugs into the InfiniBand midlayer

mlx4_en

A 10/40GigE driver under drivers/net/ethernet/mellanox/mlx4 that handles Ethernet specific functions and plugs into the netdev mid-layer

1.3.2 mlx5 Driver

`mlx5` is the low level driver implementation for the Connect-IB™ adapters designed by Mellanox Technologies. Connect-IB™ operates as an InfiniBand adapter. The mlx5 driver is comprised of the following kernel modules:

mlx5_core

Acts as a library of common functions (e.g. initializing the device after reset) required by the Connect-IB™ adapter card.

mlx5_ib

Handles InfiniBand-specific functions and plugs into the InfiniBand midlayer.

libmlx5

libmlx5 is the provider library that implements hardware specific user-space functionality. If there is no compatibility between the firmware and the driver, the driver will not load and a message will be printed in the dmesg.

The following are the Libmlx5 environment variables:

- MLX5_FREEZE_ON_ERROR_CQE
  - Causes the process to hang in a loop when completion with error which is not flushed with error or retry exceeded occurs/
  - Otherwise disabled
- MLX5_POST_SEND_PREFER_BF
  - Configures every work request that can use blue flame will use blue flame
  - Otherwise - blue flame depends on the size of the message and inline indication in the packet
- MLX5_SHUT_UP_BF
  - Disables blue flame feature
  - Otherwise - do not disable
- MLX5_SINGLE_THREADED
  - All spinlocks are disabled
  - Otherwise - spinlocks enabled
  - Used by applications that are single threaded and would like to save the overhead of taking spinlocks.
- MLX5_CQE_SIZE
  - 64 - completion queue entry size is 64 bytes (default)
  - 128 - completion queue entry size is 128 bytes

MLX5_SCATTER_TO_CQE

- Small buffers are scattered to the completion queue entry and manipulated by the driver. Valid for RC transport.
- Default is 1, otherwise disabled.

1.3.3 Mid-layer Core

Core services include: management interface (MAD), connection manager (CM) interface, and Subnet Administrator (SA) interface. The stack includes components for both user-mode and kernel applications. The core services run in the kernel and expose an interface to user-mode for verbs, CM and management.

1.3.4 ULPs

IPoIB

The IP over IB (IPoIB) driver is a network interface implementation over InfiniBand. IPoIB encapsulates IP datagrams over an InfiniBand connected or datagram transport service. IPoIB pre-appends the IP datagrams with an encapsulation header, and sends the outcome over the InfiniBand transport service. The transport service is Unreliable Datagram (UD) by default, but it may also be configured to be Reliable Connected (RC). The interface supports unicast, multicast and broadcast. For details, see Chapter 4.4, “IP over InfiniBand”.

iSER

iSCSI Extensions for RDMA (iSER) extends the iSCSI protocol to RDMA. It permits data to be transferred directly into and out of SCSI buffers without intermediate data copies. For further information, please refer to Chapter 4.2, “iSCSI Extensions for RDMA (iSER)”.

SRP

SCSI RDMA Protocol (SRP) is designed to take full advantage of the protocol offload and RDMA features provided by the InfiniBand architecture. SRP allows a large body of SCSI software to be readily used on InfiniBand architecture. The SRP driver—known as the SRP Initiator—differs from traditional low-level SCSI drivers in Linux. The SRP Initiator does not control a local HBA; instead, it controls a connection to an I/O controller—known as the SRP Target—to provide access to remote storage devices across an InfiniBand fabric. The SRP Target resides in an I/O unit and provides storage services. See Chapter 4.1, “SCSI RDMA Protocol” and Appendix A, “SRP Target Driver”.

uDAPL

User Direct Access Programming Library (uDAPL) is a standard API that promotes data center application data messaging performance, scalability, and reliability over RDMA interconnects: InfiniBand and RoCE. The uDAPL interface is defined by the DAT collaborative.

This release of the uDAPL reference implementation package for both DAT 1.2 and 2.0 specification is timed to coincide with OFED release of the Open Fabrics (www.openfabrics.org) software stack.

For more information about the DAT collaborative, go to the following site:

http://www.datcollaborative.org

1.3.5 MPI

Message Passing Interface (MPI) is a library specification that enables the development of parallel software libraries to utilize parallel computers, clusters, and heterogeneous networks. Mellanox OFED includes the following MPI implementations over InfiniBand:

- Open MPI – an open source MPI-2 implementation by the Open MPI Project
- OSU MVAPICH – an MPI-1 implementation by Ohio State University

Mellanox OFED also includes MPI benchmark tests such as OSU BW/LAT, Intel MPI Benchmark, and Presta.

1.3.6 InfiniBand Subnet Manager

All InfiniBand-compliant ULPs require a proper operation of a Subnet Manager (SM) running on the InfiniBand fabric, at all times. An SM can run on any node or on an IB switch. OpenSM is an InfiniBand-compliant Subnet Manager, and it is installed as part of Mellanox OFED.[1] See Chapter 8, “OpenSM – Subnet Manager”.

1.3.7 Diagnostic Utilities

Mellanox OFED includes the following two diagnostic packages for use by network and datacenter managers:

- ibutils – Mellanox Technologies diagnostic utilities
- infiniband-diags – OpenFabrics Alliance InfiniBand diagnostic tools

1.3.8 Mellanox Firmware Tools

The Mellanox Firmware Tools (MFT) package is a set of firmware management tools for a single InfiniBand node. MFT can be used for:

- Generating a standard or customized Mellanox firmware image
- Querying for firmware information
- Burning a firmware image to a single InfiniBand node

MFT includes the following tools:

- mlxburn - provides the following functions:
  - Generation of a standard or customized Mellanox firmware image for burning—in .bin (binary) or .img format
  - Burning an image to the Flash/EEPROM attached to a Mellanox HCA or switch device
  - Querying the firmware version loaded on an HCA board
  - Displaying the VPD (Vital Product Data) of an HCA board
- flint

  This tool burns a firmware binary image or an expansion ROM image to the Flash device of a Mellanox network adapter/bridge/switch device. It includes query functions to the burnt firmware image and to the binary image file.
- spark

1. OpenSM is disabled by default. See Chapter 8, “OpenSM – Subnet Manager” for details on enabling it.

This tool burns a firmware binary image to the EEPROM(s) attached to an InfiniScaleIII® switch device. It includes query functions to the burnt firmware image and to the binary image file. The tool accesses the EEPROM and/or switch device via an I2C-compatible interface or via vendor-specific MADs over the InfiniBand fabric (In-Band tool).

- Debug utilities

  A set of debug utilities (e.g., itrace, mstdump, isw, and i2c)

For additional details, please refer to the MFT User's Manual `docs/`.

1.4 Quality of Service

Quality of Service (QoS) requirements stem from the realization of I/O consolidation over an IB and Eth network. As multiple applications and ULPs share the same fabric, a means is needed to control their use of network resources.

QoS over Mellanox OFED for Linux is discussed in Chapter 8, "OpenSM – Subnet Manager".

1.5 RDMA over Converged Ethernet (RoCE)

RoCE allows InfiniBand (IB) transport applications to work over Ethernet network. RoCE encapsulates the InfiniBand transport and the GRH headers in Ethernet packets bearing a dedicated ether type (0x8195). Thus, any VERB application that works in an InfiniBand fabric can work in an Ethernet fabric as well.

RoCE is enabled only for drivers that support VPI (currently, only mlx4). When working with RDMA applications over Ethernet link layer the following points should be noted:

- The presence of a Subnet Manager (SM) is not required in the fabric. Thus, operations that require communication with the SM are managed in a different way in RoCE. This does not affect the API but only the actions such as joining multicast group, that need to be taken when using the API
- Since LID is a layer 2 attribute of the InfiniBand protocol stack, it is not set for a port and is displayed as zero when querying the port
- With RoCE, the alternate path is not set for RC QP and therefore APM is not supported
- Since the SM is not present, querying a path is impossible. Therefore, the path record structure must be filled with the relevant values before establishing a connection. Hence, it is recommended working with RDMA-CM to establish a connection as it takes care of filling the path record structure
- The GID table for each port is populated with N+1 entries where N is the number of IP addresses that are assigned to all network devices associated with the port including VLAN devices, alias devices and bonding masters. The only exception to this rule is a bonding master of a slave in a DOWN state. In that case, a matching GID to the IP address of the master will not be present in the GID table of the slave's port
- The first entry in the GID table (at index 0) for each port is always present and equal to the link local IPv6 address of the net device that is associated with the port. Note that even if the link local IPv6 address is not set, index 0 is still populated.

- GID format can be of 2 types, IPv4 and IPv6. IPv4 GID is a IPv4-mapped IPv6 address[1] while IPv6 GID is the IPv6 address itself
- VLAN tagged Ethernet frames carry a 3-bit priority field. The value of this field is derived from the IB SL field by taking the 3 least significant bits of the SL field
- RoCE traffic is not shown in the associated Ethernet device's counters since it is offloaded by the hardware and does not go through Ethernet network driver. RoCE traffic is counted in the same place where InfiniBand traffic is counted; /sys/class/infiniband/<device>/ports/<port number>/counters/

1.5.1 Configuring DAPL over RoCE

The default dat.conf file which contains entries for the DAPL devices, does not contain entries for the DAPL over RDMA_CM over RoCE devices.

➢ ***To add the missing entries:***

Step 1. Run the ibdev2netdev utility to see all the associations between the Ethernet devices and the IB devices/ports.

Step 2. Add a new entry line according to the format below to the dat.conf file for each output line of the ibdev2netdev utility.

<IA Name> u2.0 nonthreadsafe default libdaplofa.so.2 dapl.2.0 "<ethX> <port>" ""

Parameter	Description	Example
<IA Name>	The device's IA name. The name must be unique.	ofa-v2-ethx
<ethX>	The associated Ethernet device used by RoCE.	eth3
<port>	The port number.	1

The following is an example of the ibdev2netdev utility's output and the entries added per each output line:

```
sw419:~ # ibdev2netdev mlx4_0 port 2 <===> eth2 mlx4_0 port 1 <===> eth3
ofa-v2-eth2 u2.0 nonthreadsafe default libdaplofa.so.2 dapl.2.0 "eth2 2" ""
ofa-v2-eth3 u2.0 nonthreadsafe default libdaplofa.so.2 dapl.2.0 "eth3 1" ""
```

1. For the IPv4 address A.B.C.D the corresponding IPv4-mapped IPv6 address is ::ffff.A.B.C.D

1.5.2 A Detailed Example

This section provides information of how to use InfiniBand over Ethernet (RoCE).

1.5.2.1 Installing and Loading the Driver

➢ ***To install and load the driver:***

Step 1. Install MLNX_OFED (See section Section 2.3, “Installing Mellanox OFED”, on page 35 for details on installation.)

RoCE is installed as part of mlx4 and mlx4_en and other modules upon driver’s installation.

The list of the modules that will be loaded automatically upon boot can be found in the configuration file /etc/infiniband/openib.conf.

Step 2. Display the existing MLNX_OFED version.

```
# ibv_devinfo
hca_id: mlx4_0
        transport: InfiniBand (0)
        fw_ver: 2.31.5050
        node_guid: 0002:c903:0008:e810
        sys_image_guid: 0002:c903:0008:e813
        vendor_id: 0x02c9
        vendor_part_id: 4099
        hw_ver: 0xB0
        board_id: MT_1060110019
        phys_port_cnt: 2
             port: 1
                   state: PORT_INIT (2)
                   max_mtu: 2048 (4)
                   active_mtu: 2048 (4)
                   sm_lid: 0
                   port_lid: 0
                   port_lmc: 0x00
                   link_layer: IB
             port: 2
                   state: PORT_ACTIVE (4)
                   max_mtu: 2048 (4)
                   active_mtu: 1024 (3)
                   sm_lid: 0
                   port_lid: 0
                   port_lmc: 0x00
                   link_layer: Ethernet
#
```

Notes about the output above:

The ports' states are:
• Port 1 - InfiniBand, is in PORT_INIT state
• Port 2 - Ethernet is in PORT_ACTIVE state
	The port state can also be obtained by running the following commands:
	`# cat /sys/class/infiniband/mlx4_0/ports/1/state 2: INIT`
	`# cat /sys/class/infiniband/mlx4_0/ports/2/state 4: ACTIVE`
	`#`

The link_layer parameter shows that:
• Port 1 is InfiniBand
• Port 2 is Ethernet
Nevertheless, Port 2 appears in the list of the HCA's ports.
	The link_layer of the two ports can also be obtained by running the following commands:
	`# cat /sys/class/infiniband/mlx4_0/ports/1/link_layer InfiniBand`
	`# cat /sys/class/infiniband/mlx4_0/ports/2/link_layer Ethernet`
	`#`

The fw_ver parameter shows that the firmware version is 2.31.5050.	The firmware version can also be obtained by running the following commands:
	`# cat /sys/class/infiniband/mlx4_0/fw_ver 2.31.5050`
	`#`

Although the InfiniBand over Ethernet's Port MTU is 2K byte at maximum, the actual MTU cannot exceed the mlx4_en interface's MTU. Since the mlx4_en interface's MTU is typically 1560, port 2 will run with MTU of 1K. Please note that RoCE's MTU are subject to InfiniBand MTU restrictions. The RoCE's MTU values are, 256 byte, 512 byte, 1024 byte and 2K. In general RoCE MTU is the largest power of 2 that is still lower than mlx4_en interface MTU.	-

1.5.2.2 Associating InfiniBand Ports to Ethernet Ports

Since both RoCE and mlx4_en use the Ethernet port of the adapter, one of the drivers must control the port state. In the example above, the mlx4_en driver controls the port's state. The mlx4_ib driver holds a reference to the mlx4_en net device for getting notifications about the state of the port, as well as using the mlx4_en driver to resolve IP addresses to MAC that are required for address vector creation. However, RoCE traffic does not go through the mlx4_en driver, it is completely offloaded by the hardware.

```
# ibdev2netdev
mlx4_0 port 2 <===> eth2
mlx4_0 port 1 <===> ib0
#
```

1.5.2.3 Configuring an IP Address to the mlx4_en Interface

➢ ***To configure an IP address to the mlx4_en interface:***

Step 1. Configure an IP address to the mlx4_en interface on both sides of the link.

```
# ifconfig eth2 20.4.3.220
# ifconfig eth2
eth2              Link encap:Ethernet HWaddr 00:02:C9:08:E8:11
                  inet addr:20.4.3.220 Bcast:20.255.255.255 Mask:255.0.0.0
                  UP BROADCAST MULTICAST MTU:1500 Metric:1
                  RX packets:0 errors:0 dropped:0 overruns:0 frame:0
                  TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
                  collisions:0 txqueuelen:1000
                  RX bytes:0 (0.0 b) TX bytes:0 (0.0 b)
```

Step 2. Make sure that ping is working.

```
# ping 20.4.3.219
PING 20.4.3.219 (20.4.3.219) 56(84) bytes of data.
64 bytes from 20.4.3.219: icmp_seq=1 ttl=64 time=0.873 ms
64 bytes from 20.4.3.219: icmp_seq=2 ttl=64 time=0.198 ms
64 bytes from 20.4.3.219: icmp_seq=3 ttl=64 time=0.167 ms
--- 20.4.3.219 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2000ms
rtt min/avg/max/mdev = 0.167/0.412/0.873/0.326 ms
```

1.5.2.4 Inspecting the GID Table

➢ ***To inspect the GID table:***

```
# cat /sys/class/infiniband/mlx4_0/ports/2/gids/0
fe80:0000:0000:0000:0202:c9ff:fe08:e811
#
# cat /sys/class/infiniband/mlx4_0/ports/2/gids/1
0000:0000:0000:0000:0000:0000:0000:0000
#
```

According to the output, we currently have one entry only.

1.5.2.5 Running an - ibv_rc_pingpong Test

Step 1. Start the server.

```
# ibv_rc_pingpong -g 0 -i 2
local address: LID 0x0000, QPN 0x00004f, PSN 0x3315f6, GID fe80::202:c9ff:fe08:e799
remote address: LID 0x0000, QPN 0x04004f, PSN 0x2cdede, GID fe80::202:c9ff:fe08:e811
8192000 bytes in 0.01 seconds = 4730.13 Mbit/sec
1000 iters in 0.01 seconds = 13.85 usec/iter
#
```

Step 2. Start the client.

```
# ibv_rc_pingpong -g 0 -i 2 sw419
local address: LID 0x0000, QPN 0x04004f, PSN 0x2cdede, GID fe80::202:c9ff:fe08:e811
remote address: LID 0x0000, QPN 0x00004f, PSN 0x3315f6, GID fe80::202:c9ff:fe08:e799
8192000 bytes in 0.01 seconds = 4787.84 Mbit/sec
1000 iters in 0.01 seconds = 13.69 usec/iter
#
```

1.5.2.6 Adding VLANs

➢ ***To add VLANs:***

Step 1. Make sure that the 8021.q module is loaded.

```
# modprobe 8021q
```

Step 2. Add VLAN.

```
# vconfig add eth2 7
Added VLAN with VID == 7 to IF -:eth2:-
#
```

Step 3. Configure an IP address.

```
# ifconfig eth2.7 7.4.3.220
```

Step 4. Examine the GID table.

```
# cat /sys/class/infiniband/mlx4_0/ports/2/gids/0
fe80:0000:0000:0000:0202:c9ff:fe08:e811
#
# cat /sys/class/infiniband/mlx4_0/ports/2/gids/1
fe80:0000:0000:0000:0202:c900:0708:e811
```

1.5.2.7 Running an - ibv_rc_pingpong Test on the VLAN

Step 1. Start the server.

```
# ibv_rc_pingpong -g 1 -i 2
local address: LID 0x0000, QPN 0x04004f, PSN 0xbdde2c, GID fe80::202:c900:708:e799
remote address: LID 0x0000, QPN 0x08004f, PSN 0xc9d800, GID fe80::202:c900:708:e811
8192000 bytes in 0.01 seconds = 4824.50 Mbit/sec
1000 iters in 0.01 seconds = 13.58 usec/iter
```

Step 2. Start the client.

```
# ibv_rc_pingpong -g 1 -i 2 sw419
local address: LID 0x0000, QPN 0x08004f, PSN 0xc9d800, GID fe80::202:c900:708:e811
remote address: LID 0x0000, QPN 0x04004f, PSN 0xbdde2c, GID fe80::202:c900:708:e799
8192000 bytes in 0.01 seconds = 4844.83 Mbit/sec
1000 iters in 0.01 seconds = 13.53 usec/iter
```

1.5.2.8 Defining Ethernet Priority (PCP in 802.1q Headers)

Step 1. Define ethernet priority on the server.

```
# ibv_rc_pingpong -g 1 -i 2 -l 4
local address: LID 0x0000, QPN 0x1c004f, PSN 0x9daf6c, GID fe80::202:c900:708:e799
remote address: LID 0x0000, QPN 0x1c004f, PSN 0xb0a49b, GID fe80::202:c900:708:e811
8192000 bytes in 0.01 seconds = 4840.89 Mbit/sec
1000 iters in 0.01 seconds = 13.54 usec/iter
```

Step 2. Define ethernet priority on the client.

```
# ibv_rc_pingpong -g 1 -i 2 -l 4 sw419
local address: LID 0x0000, QPN 0x1c004f, PSN 0xb0a49b, GID fe80::202:c900:708:e811
remote address: LID 0x0000, QPN 0x1c004f, PSN 0x9daf6c, GID fe80::202:c900:708:e799
8192000 bytes in 0.01 seconds = 4855.96 Mbit/sec
1000 iters in 0.01 seconds = 13.50 usec/iter
```

1.5.2.9 Using rdma_cm Tests

Step 1. Use rdma_cm test on the server.

```
# ucmatose
cmatose: starting server
initiating data transfers
completing sends
receiving data transfers
data transfers complete
cmatose: disconnecting
disconnected
test complete
return status 0
#
```

Step 2. Use rdma_cm test on the client.

```
# ucmatose -s 20.4.3.219
cmatose: starting client
cmatose: connecting
receiving data transfers
sending replies
data transfers complete
test complete
return status 0
#
```

This server-client run is without PCP or VLAN because the IP address used does not belong to a VLAN interface. If you specify a VLAN IP address, then the traffic should go over VLAN.

1.5.2.10 Type Of Service (TOS)

The TOS field for rdma_cm sockets can be set using the rdma_set_option() API, just as it is set for regular sockets. If a TOS is not set, the default value (0) is used. Within the rdma_cm kernel driver, the TOS field is converted into an SL field. The conversion formula is as follows:

- SL = TOS >> 5 (e.g., take the 3 most significant bits of the TOS field)

In the hardware driver, the SL field is converted into PCP by the following formula:

- PCP = SL & 7 (take the 3 least significant bits of the TOS field)

SL affects the PCP only when the traffic goes over tagged VLAN frames.

2 Installation

This chapter describes how to install and test the Mellanox OFED for Linux package on a single host machine with Mellanox InfiniBand and/or Ethernet adapter hardware installed.

2.1 Hardware and Software Requirements

Table 1 - Software and Hardware Requirements

Requirements	Description
Platforms	A server platform with an adapter card based on one of the following Mellanox Technologies' InfiniBand HCA devices: • MT25408 ConnectX®-2 (VPI, IB, EN) (firmware: fw-ConnectX2) • MT27508 ConnectX®-3 (VPI, IB, EN) (firmware: fw-ConnectX3) • MT4103 ConnectX®-3 Pro (VPI, IB, EN) (firmware: fw-ConnectX3Pro) • MT4113 Connect-IB® (IB) (firmware: fw-Connect-IB) For the list of supported architecture platforms, please refer to the Mellanox OFED Release Notes file.
Required Disk Space for Installation	1GB
Device ID	For the latest list of device IDs, please visit Mellanox website.
Operating System	Linux operating system. For the list of supported operating system distributions and kernels, please refer to the Mellanox OFED Release Notes file.
Installer Privileges	The installation requires administrator privileges on the target machine.

2.2 Downloading Mellanox OFED

Step 3. Verify that the system has a Mellanox network adapter (HCA/NIC) installed by ensuring that you can see ConnectX or InfiniHost entries in the display.

The following example shows a system with an installed Mellanox HCA:

```
# lspci -v | grep Mellanox
06:00.0 Network controller: Mellanox Technologies MT27500 Family [ConnectX-3]
Subsystem: Mellanox Technologies Device 0024
```

Step 4. Download the ISO image to your host.

The image's name has the format `MLNX_OFED_LINUX-<ver>-<OS label><CPU arch>.iso`. You can download it from http://www.mellanox.com > Products > Software> InfiniBand Drivers.

Step 5. Use the md5sum utility to confirm the file integrity of your ISO image. Run the following command and compare the result to the value provided on the download page.

```
host1$ md5sum MLNX_OFED_LINUX-<ver>-<OS label>.iso
```

2.3 Installing Mellanox OFED

The installation script, `mlnxofedinstall`, performs the following:

- Discovers the currently installed kernel
- Uninstalls any software stacks that are part of the standard operating system distribution or another vendor's commercial stack
- Installs the MLNX_OFED_LINUX binary RPMs (if they are available for the current kernel)
- Identifies the currently installed InfiniBand and Ethernet network adapters and automatically[1] upgrades the firmware

2.3.1 Pre-installation Notes

- The installation script removes all previously installed Mellanox OFED packages and re-installs from scratch. You will be prompted to acknowledge the deletion of the old packages.

Pre-existing configuration files will be saved with the extension ".conf.rpmsave".

- If you need to install Mellanox OFED on an entire (homogeneous) cluster, a common strategy is to mount the ISO image on one of the cluster nodes and then copy it to a shared file system such as NFS. To install on all the cluster nodes, use cluster-aware tools (such as pdsh).
- If your kernel version does not match with any of the offered pre-built RPMs, you can add your kernel version by using the "`mlnx_add_kernel_support.sh`" script located under the docs/ directory.

On Redhat and SLES distributions with errata kernel installed there is no need to use the mlnx_add_kernel_support.sh script. The regular installation can be performed and weak-updates mechanism will create symbolic links to the MLNX_OFED kernel modules.

Usage:

```
mlnx_add_kernel_support.sh -m|--mlnx_ofed <path to MLNX_OFED directory> [--make-iso|--
make-tgz]

[--make-iso] Create MLNX_OFED ISO image.
[--make-tgz] Create MLNX_OFED tarball. (Default)
[-t|--tmpdir <local work dir>]
[--kmp] Enable KMP format if supported.
[-k | --kernel] <kernel version> Kernel version to use.
[-s | --kernel-sources] <path to the kernel sources> Path to kernel headers.
[-v|--verbose]
[-n|--name] Name of the package to be created.
[-y|--yes] Answer "yes" to all questions
[--force] Force removing packages that depends on MLNX_OFED
```

1. The firmware will not be updated if you run the install script with the '--without-fw-update' option.

Example

The following command will create a MLNX_OFED_LINUX ISO image for RedHat 6.3 under the /tmp directory.

```
# ./MLNX_OFED_LINUX-2.2-1.0.0-rhel6.3-x86_64/mlnx_add_kernel_support.sh -m /tmp/
MLNX_OFED_LINUX-2.2-1.0.0-rhel6.3-x86_64/ --make-tgz
Note: This program will create MLNX_OFED_LINUX TGZ for rhel6.3 under /tmp directory.
All Mellanox, OEM, OFED, or Distribution IB packages will be removed.
Do you want to continue?[y/N]:y
See log file /tmp/mlnx_ofed_iso.21642.log

Building OFED RPMs. Please wait...
Removing OFED RPMs...
Created /tmp/MLNX_OFED_LINUX-2.2-1.0.0-rhel6.3-x86_64-ext.tgz
```

The "mlnx_add_kernel_support.sh" script can be executed directly from the mlnxofedinstall script. For further information, please see '--add-kernel-support' option below.

2.3.2 Installation Script

Mellanox OFED includes an installation script called `mlnxofedinstall`. Its usage is described below. You will use it during the installation procedure described in Section 2.3.3, "Installation Procedure", on page 39.

Usage

```
./mnt/mlnxofedinstall [OPTIONS]
```

Options

-c|--config <packages config_file>	Example of the configuration file can be found under docs
-n|--net <network config_file>	Example of the network configuration file can be found under docs
-k|--kernel-version <kernel version>	Use provided kernel version instead of 'uname -r'
-p|--print-available	Print available packages for current platform and create corresponding ofed.conf file
--with-32bit	Install 32-bit libraries
--without-32bit	Skip 32-bit libraries installation (Default)
--without-depcheck	Skip Distro's libraries check
--without-fw-update	Skip firmware update
--fw-update-only	Update firmware. Skip driver installation

--force-fw-update	Force firmware update
--force	Force installation
--all|--hpc|--basic|--msm	Install all, hpc, basic or Mellanox Subnet manager packages correspondingly
--vma|--vma-vpi	Install packages required by VMA to support VPI
--vma-eth	Install packages required by VMA to work over Ethernet
--with-vma	Set configuration for VMA use (to be used with any installation parameter).
--guest	Install packages required by guest os
--hypervisor	Install packages required by hypervisor os
-v|-vv|-vvv	Set verbosity level
--umad-dev-rw	Grant non root users read/write permission for umad devices instead of default
--umad-dev-na	Prevent from non root users read/write access for umad devices. Overrides '--umad-dev-rw'
--enable-affinity	Run mlnx_affinity script upon boot
--disable-affinity	Disable mlnx_affinity script (Default)
--enable-sriov	Burn SR-IOV enabled firmware - Note: Enable/Disable of SRIOV in a non-volatile configuration through uEFI and/or tool will override this flag.
--add-kernel-support	Add kernel support (Run mlnx_add_kernel_support.sh)
--skip-distro-check	Do not check MLNX_OFED vs Distro matching
--hugepages-overcommit	Setting 80% of MAX_MEMORY as overcommit for huge page allocation
-q	Set quiet - no messages will be printed
--without-<package>	Do not install package
--with-fabric-collector	Install fabric-collector package.

2.3.2.1 mlnxofedinstall Return Codes

Table 2 lists the `mlnxofedinstall` script return codes and their meanings.

Table 2 - mlnxofedinstall Return Codes

Return Code	Meaning
0	The Installation ended successfully
1	The installation failed
2	No firmware was found for the adapter device
22	Invalid parameter
28	Not enough free space
171	Not applicable to this system configuration. This can occur when the required hardware is not present on the system.
172	Prerequisites are not met. For example, missing the required software installed or the hardware is not configured correctly.
173	Failed to start the `mst` driver

2.3.3 Installation Procedure

Step 1. Login to the installation machine as root.

Step 2. Mount the ISO image on your machine

```
host1# mount -o ro,loop MLNX_OFED_LINUX-<ver>-<OS label>-<CPU arch>.iso /mnt
```

Step 3. Run the installation script.

```
Logs dir: /tmp/MLNX_OFED_LINUX-2.2-0.0.9.10694.logs
This program will install the MLNX_OFED_LINUX package on your machine.
Note that all other Mellanox, OEM, OFED, or Distribution IB packages will be removed.
Uninstalling the previous version of MLNX_OFED_LINUX

Starting MLNX_OFED_LINUX-2.2-0.0.9 installation ...

Installing mlnx-ofa_kernel RPM
Preparing...                ##################################################
mlnx-ofa_kernel             ##################################################
Installing kmod-mlnx-ofa_kernel 2.2 RPM
Preparing...                ##################################################
kmod-mlnx-ofa_kernel        ##################################################
Installing mlnx-ofa_kernel-devel RPM
Preparing...                ##################################################
mlnx-ofa_kernel-devel       ##################################################
Installing kmod-kernel-mft-mlnx 3.6.0 RPM
Preparing...                ##################################################
kmod-kernel-mft-mlnx        ##################################################
Installing knem-mlnx RPM
Preparing...                ##################################################
knem-mlnx                   ##################################################
Installing kmod-knem-mlnx 1.1.1.90mlnx RPM
Preparing...                ##################################################
kmod-knem-mlnx              ##################################################
Installing ummunotify-mlnx RPM
Preparing...                ##################################################
ummunotify-mlnx             ##################################################
Installing kmod-ummunotify-mlnx 1.0 RPM
Preparing...                ##################################################
kmod-ummunotify-mlnx        ##################################################
Installing kmod-iser 1.2 RPM
Preparing...                ##################################################
kmod-iser                   ##################################################
Installing kmod-srp 1.3.2 RPM
Preparing...                ##################################################
kmod-srp                    ##################################################
Installing mpi-selector RPM
Preparing...                ##################################################
mpi-selector                ##################################################
```

```
Installing user level RPMs:
Preparing...                ##################################################
ofed-scripts                ##################################################
Preparing...                ##################################################
libibverbs                  ##################################################
Preparing...                ##################################################
libibverbs-devel            ##################################################
Preparing...                ##################################################
libibverbs-devel-static     ##################################################
Preparing...                ##################################################
libibverbs-utils            ##################################################
Preparing...                ##################################################
libmlx4                     ##################################################
Preparing...                ##################################################
libmlx4-devel               ##################################################
Preparing...                ##################################################
libmlx5                     ##################################################
Preparing...                ##################################################
libmlx5-devel               ##################################################
Preparing...                ##################################################
libibcm                     ##################################################
Preparing...                ##################################################
libibcm-devel               ##################################################
Preparing...                ##################################################
libibumad                   ##################################################
Preparing...                ##################################################
libibumad-devel             ##################################################
Preparing...                ##################################################
libibumad-static            ##################################################
Preparing...                ##################################################
libibmad                    ##################################################
Preparing...                ##################################################
libibmad-devel              ##################################################
Preparing...                ##################################################
libibmad-static             ##################################################
Preparing...                ##################################################
ibsim                       ##################################################
Preparing...                ##################################################
ibacm                       ##################################################
Preparing...                ##################################################
librdmacm                   ##################################################
Preparing...                ##################################################
librdmacm-utils             ##################################################
Preparing...                ##################################################
librdmacm-devel             ##################################################
Preparing...                ##################################################
opensm-libs                 ##################################################
Preparing...                ##################################################
opensm                      ##################################################
Preparing...                ##################################################
opensm-devel                ##################################################
Preparing...                ##################################################
opensm-static               ##################################################
```

```
Preparing...                ########################################################
dapl                        ########################################################
Preparing...                ########################################################
dapl-devel                  ########################################################
Preparing...                ########################################################
dapl-devel-static           ########################################################
Preparing...                ########################################################
dapl-utils                  ########################################################
Preparing...                ########################################################
perftest                    ########################################################
Preparing...                ########################################################
mstflint                    ########################################################
Preparing...                ########################################################
mft                         ########################################################
Preparing...                ########################################################
srptools                    ########################################################
Preparing...                ########################################################
rds-tools                   ########################################################
Preparing...                ########################################################
rds-devel                   ########################################################
Preparing...                ########################################################
ibutils2                    ########################################################
Preparing...                ########################################################
ibutils                     ########################################################
Preparing...                ########################################################
cc_mgr                      ########################################################
Preparing...                ########################################################
dump_pr                     ########################################################
Preparing...                ########################################################
ar_mgr                      ########################################################
Preparing...                ########################################################
ibdump                      ########################################################
Preparing...                ########################################################
infiniband-diags            ########################################################
Preparing...                ########################################################
infiniband-diags-compat     ########################################################
Preparing...                ########################################################
qperf                       ########################################################
Preparing...                ########################################################
fca                         ########################################################
Preparing...                ########################################################
mxm                         ########################################################
Preparing...                ########################################################
openmpi                     ########################################################
Preparing...                ########################################################
openmpi                     ########################################################
Preparing...                ########################################################
bupc                        ########################################################
Preparing...                ########################################################
infinipath-psm              ########################################################
Preparing...                ########################################################
infinipath-psm-devel        ########################################################
```

```
Preparing...                ########################################################
mvapich2                    ########################################################
Preparing...                ########################################################
hcoll                       ########################################################
Preparing...                ########################################################
libibprof                   ########################################################
Preparing...                ########################################################
mlnxofed-docs               ########################################################
Preparing...                ########################################################
mpitests_mvapich2__2_0rc1   ########################################################
Preparing...                ########################################################
mpitests_openmpi__1_6_5     ########################################################
Preparing...                ########################################################
mpitests_openmpi__1_8       ########################################################
Device (06:00.0):
       06:00.0 Network controller: Mellanox Technologies MT27500 Family [ConnectX-3]
        Link Width: 8x
        PCI Link Speed: 5Gb/s
Installation finished successfully.

Attempting to perform Firmware update...
Querying Mellanox devices firmware ...

Device #1:
----------
  Device Type:      ConnectX3Pro
  Part Number:      MCX354A-FCC_Ax
  Description:     ConnectX-3 Pro VPI adapter card; dual-port QSFP; FDR IB (56Gb/s)
and 40GigE;PCIe3.0 x8 8GT/s;RoHS R6
  PSID:             MT_1090111019
  PCI Device Name:  0000:05:00.0
  Versions:         Current        Available
     FW             2.30.8000      2.31.5000
     PXE            3.4.0224       3.4.0224

Status:        Update required
Found 1 device(s) requiring firmware update...

Device #1: Updating FW ... Done

A restart is needed for updates to take effect.
Log File: /tmp/MLNX_OFED_LINUX-2.2-0.0.9.10694.logs/fw_update.log
To load the new driver, run:
/etc/init.d/openibd restart
```

In case your machine has the latest firmware, no firmware update will occur and the installation script will print at the end of installation a message similar to the following:

```
Device #1:
----------
  Device Type:      ConnectX3Pro
  Part Number:      MCX354A-FCC_Ax
  Description:      ConnectX-3 Pro VPI adapter card; dual-port
QSFP; FDR IB (56Gb/s) and 40GigE;PCIe3.0 x8 8GT/s;RoHS R6
  PSID:             MT_1090111019
  PCI Device Name:  0000:05:00.0
  Versions:         Current        Available
     FW             2.31.5000      2.31.5000
     PXE            3.4.0224       3.4.0224
  Status:        Up to date
```

In case your machine has an unsupported network adapter device, no firmware update will occur and the error message below will be printed. Please contact your hardware vendor for help on firmware updates.

Error message:

```
Device #1:
----------
  Device:         0000:05:00.0
  Part Number:
  Description:
  PSID:           MT_0DB0110010
  Versions:       Current        Available
     FW           2.9.1000       N/A
  Status:         No matching image found
```

Step 4. Reboot the machine **if** the installation script performed firmware updates to your network adapter hardware. Otherwise, restart the driver by running: "`/etc/init.d/openibd restart`"

Note: The script adds the following lines to `/etc/security/limits.conf` for the user-space components such as MPI:

```
* soft memlock unlimited
* hard memlock unlimited
```

These settings set the amount of memory that can be pinned by a user space application to unlimited. If desired, tune the value unlimited to a specific amount of RAM.

Note: For your machine to be part of the InfiniBand/VPI fabric, a Subnet Manager must be running on one of the fabric nodes. At this point, Mellanox OFED for Linux has already installed the OpenSM Subnet Manager on your machine. For details on starting OpenSM, see Chapter 8, “OpenSM – Subnet Manager”.

Step 5. (InfiniBand only) Run the `hca_self_test.ofed` utility to verify whether or not the InfiniBand link is up. The utility also checks for and displays additional information such as

- HCA firmware version
- Kernel architecture

- Driver version
- Number of active HCA ports along with their states
- Node GUID

Note: For more details on `hca_self_test.ofed`, see the file `hca_self_test.readme` under `docs/`.

```
# hca_self_test.ofed

---- Performing Adapter Device Self Test ----
Number of CAs Detected ................. 1
PCI Device Check ....................... PASS
Kernel Arch ............................ x86_64
Host Driver Version .................... MLNX_OFED_LINUX-2.2-1.0.0 (OFED-2.2-1.0.0):
3.0.76-0.11-default
Host Driver RPM Check .................. PASS
Firmware on CA #0 VPI .................. v2.31.5000
Firmware Check on CA #0 (VPI) .......... PASS
Host Driver Initialization ............. PASS
Number of CA Ports Active .............. 1
Port State of Port #1 on CA #0 (VPI)..... UP 4X FDR (InfiniBand)
Port State of Port #2 on CA #0 (VPI)..... UP 4X FDR (InfiniBand)
Error Counter Check on CA #0 (VPI)...... PASS
Kernel Syslog Check .................... PASS
Node GUID on CA #0 (VPI) ............... 00:02:c9:03:00:30:0e:60
------------------ DONE ---------------------
```

After the installer completes, information about the Mellanox OFED installation such as prefix, kernel version, and installation parameters can be retrieved by running the command `/etc/infiniband/info`.

2.3.4 Installation Results

Software

- Most of MLNX_OFED packages are installed under the “/usr” directory except for the following packages which are installed under the “/opt” directory:
  - openshmem, bupc, fca and ibutils
- The kernel modules are installed under
  - /lib/modules/`uname -r`/updates on SLES and Fedora Distributions
  - /lib/modules/`uname -r`/extra/mlnx-ofa_kernel on RHEL and other RedHat like Distributions
  - /lib/modules/`uname -r`/updates/dkms/ on Ubuntu

Firmware

- The firmware of existing network adapter devices will be updated if the following two conditions are fulfilled:

a. You run the installation script in default mode; that is, *without* the option '--without-fw-update'.

b. The firmware version of the adapter device is older than the firmware version included with the Mellanox OFED ISO image

If an adapter's Flash was originally programmed with an Expansion ROM image, the automatic firmware update will also burn an Expansion ROM image.

- In case your machine has an unsupported network adapter device, no firmware update will occur and the error message below will be printed. Please contact your hardware vendor for help on firmware updates.

  Error message:

```
-I- Querying device ...
-E- Can't auto detect fw configuration file: ...
```

2.3.5 Post-installation Notes

- Most of the Mellanox OFED components can be configured or reconfigured after the installation by modifying the relevant configuration files. See the relevant chapters in this manual for details.
- The list of the modules that will be loaded automatically upon boot can be found in the `/etc/infiniband/openib.conf` file.

2.3.6 Installation Logging

While installing MLNX_OFED, the install log for each selected package will be saved in a separate log file.

The path to the directory containing the log files will be displayed after running the installation script in the following format: `"Logs dir: /tmp/MLNX_OFED_LINUX-<version>.<PID>.logs"`.

Example:

```
Logs dir: /tmp/MLNX_OFED_LINUX-2.2-0.0.2.31701.logs
```

2.3.7 openibd Script

As of MLNX_OFED v2.2-1.0.0 the `openibd` script supports pre/post start/stop scripts:

This can be controlled by setting the variables below in the `/etc/infiniband/openibd.conf` file.

```
OPENIBD_PRE_START
OPENIBD_POST_START
OPENIBD_PRE_STOP
OPENIBD_POST_STOP
```

Example:

```
OPENIBD_POST_START=/sbin/openibd_post_start.sh
```

2.4 Updating Firmware After Installation

In case you ran the `mlnxofedinstall` script with the '`--without-fw-update`' option and now you wish to (manually) update firmware on your adapter card(s), you need to perform the following steps:

If you need to burn an Expansion ROM image, please refer to "Burning the Expansion ROM Image" on page 258

The following steps are also appropriate in case you wish to burn newer firmware that you have downloaded from Mellanox Technologies' Web site (http://www.mellanox.com > Support > Firmware Download).

Step 1. Start `mst`.

```
host1# mst start
```

Step 2. Identify your target InfiniBand device for firmware update.

1. Get the list of InfiniBand device names on your machine.

```
host1# mst status
MST modules:
------------
    MST PCI module loaded
    MST PCI configuration module loaded
    MST Calibre (I2C) module is not loaded

MST devices:
------------
/dev/mst/mt25418_pciconf0        - PCI configuration cycles access.
                                   bus:dev.fn=02:00.0 addr.reg=88
                                   data.reg=92
                                   Chip revision is: A0
/dev/mst/mt25418_pci_cr0         - PCI direct access.
                                   bus:dev.fn=02:00.0 bar=0xdef00000
                                   size=0x100000
                                   Chip revision is: A0
/dev/mst/mt25418_pci_msix0       - PCI direct access.
                                   bus:dev.fn=02:00.0 bar=0xdeefe000
                                   size=0x2000
/dev/mst/mt25418_pci_uar0        - PCI direct access.
                                   bus:dev.fn=02:00.0 bar=0xdc800000
                                   size=0x800000
```

2. Your InfiniBand device is the one with the postfix "_pci_cr0". In the example listed above, this will be `/dev/mst/mt25418_pci_cr0`.

Step 3. Burn firmware.

- Burn a firmware image from a .mlx file using the `mlxburn` utility (that is already installed on your machine)

The following command burns firmware onto the ConnectX device with the device name obtained in the example of Step 2.

```
> flint -d /dev/mst/mt25418_pci_cr0 -i fw-25408-2_1_8000-MCX353A-FCA_A1.bin burn
```

Step 4. Reboot your machine after the firmware burning is completed.

2.5 Installing MLNX_OFED using YUM

2.5.1 Setting up MLNX_OFED YUM Repository

Step 1. Log into the installation machine as root.

Step 2. Mount the ISO image on your machine.

```
# mount -o ro,loop MLNX_OFED_LINUX-<ver>-<OS label>-<CPU arch>.iso /mnt
```

You can download it from http://www.mellanox.com > Products > Software> InfiniBand Drivers.

Step 3. Download and install Mellanox Technologies GPG-KEY:

The key can be downloaded via the following link:

http://www.mellanox.com/downloads/ofed/RPM-GPG-KEY-Mellanox

```
# wget http://www.mellanox.com/downloads/ofed/RPM-GPG-KEY-Mellanox
--2014-04-20 13:52:30--  http://www.mellanox.com/downloads/ofed/RPM-GPG-KEY-Mellanox
Resolving www.mellanox.com... 72.3.194.0
Connecting to www.mellanox.com|72.3.194.0|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1354 (1.3K) [text/plain]
Saving to: ?RPM-GPG-KEY-Mellanox?

100%[==============================================>] 1,354       --.-K/s   in 0s

2014-04-20 13:52:30 (247 MB/s) - ?RPM-GPG-KEY-Mellanox? saved [1354/1354]
```

Step 4. Install the key.

```
# sudo rpm --import RPM-GPG-KEY-Mellanox
warning: rpmts_HdrFromFdno: Header V3 DSA/SHA1 Signature, key ID 6224c050: NOKEY
Retrieving key from file:///repos/MLNX_OFED/<MLNX_OFED file>/RPM-GPG-KEY-Mellanox
Importing GPG key 0x6224C050:
 Userid: "Mellanox Technologies (Mellanox Technologies - Signing Key v2) <support@mel-
lanox.com>"
 From  : /repos/MLNX_OFED/<MLNX_OFED file>/RPM-GPG-KEY-Mellanox
Is this ok [y/N]:
```

Step 5. Check that the key was successfully imported.

```
# rpm -q gpg-pubkey --qf '%{NAME}-%{VERSION}-%{RELEASE}\t%{SUMMARY}\n' | grep Mellanox
gpg-pubkey-a9e4b643-520791ba    gpg(Mellanox Technologies <support@mellanox.com>)
```

Step 6. Install the createrepo.rpm package.

```
# yum install createrepo
```

Step 7. MLNX_OFED YUM repository using the "`mlnx_create_yum_repo.sh`" script located in the downloaded MLNX_OFED package.

```
# ./mlnx_create_yum_repo.sh --mlnx_ofed /mnt --target /repos
Creating MLNX_OFED_LINUX YUM Repository under /repos...
See log file /tmp/mlnx_yum.24250.log
comps file was not provided, going to build it...
Copying RPMS...
Building YUM Repository...
Creating YUM Repository settings file at: /tmp/mlnx_ofed.repo
Done.
Copy /tmp/mlnx_ofed.repo to /etc/yum.repos.d/ to use MLNX_OFED YUM Repository.
```

Step 8. Copy the created YUM repository configuration file to `/etc/yum.repos.d/`.

```
# cp /tmp/mlnx_ofed.repo /etc/yum.repos.d/
```

Step 9. Check that the repository was successfully added.

```
# yum repolist
Loaded plugins: product-id, security, subscription-manager
This system is not registered to Red Hat Subscription Management. You can use subscrip-
tion-manager to register.
repo id           repo name                                       status
mlnx_ofed         MLNX_OFED Repository                             108
rpmforge          RHEL 6Server - RPMforge.net - dag               4,597

repolist: 8,351
```

2.5.2 Installing MLNX_OFED using the YUM Tool

After setting up the YUM repository for MLNX_OFED package, perform the following:

Step 1. View the available package groups by invoking:

```
# yum grouplist | grep MLNX_OFED
   MLNX_OFED ALL
   MLNX_OFED BASIC
   MLNX_OFED GUEST
   MLNX_OFED HYPERVISOR
   MLNX_OFED VMA
   MLNX_OFED VMA-ETH
   MLNX_OFED VMA-VPI
```

Where:

```
MLNX_OFED ALL           Installs all available packages in MLNX_OFED.
MLNX_OFED BASIC         Installs basic packages required for running Mellanox cards.
MLNX_OFED GUEST         Installs packages required by guest OS.
MLNX_OFED HYPERVISOR    Installs packages required by hypervisor OS.
MLNX_OFED VMA           Installs packages required by VMA.
MLNX_OFED VMA-ETH       Installs packages required by VMA to work over Ethernet.
MLNX_OFED VMA-VPI       Installs packages required by VMA to support VPI.
```

Step 2. Install the desired group.

```
 # yum groupinstall 'MLNX_OFED ALL'
 Loaded plugins: product-id, security, subscription-manager
 This system is not registered to Red Hat Subscription Management. You can use subscrip-
tion-manager to register.
 Setting up Group Process
 Resolving Dependencies
 --> Running transaction check
 ---> Package ar_mgr.x86_64 0:1.0-0.11.g22fff4a will be installed
 ..............
 ..............
   rds-devel.x86_64 0:2.0.6mlnx-1
   rds-tools.x86_64 0:2.0.6mlnx-1
   srptools.x86_64 0:0.0.4mlnx3-OFED.2.0.2.6.7.11.ge863cb7
 Complete!
```

2.5.3 Updating Firmware After Installation

Installing MLNX_OFED using the YUM tool does not automatically update the firmware.

To update the firmware to the version included in MLNX_OFED package, you can either:

- Run the `mlnxofedinstall` script with the `"--fw-update-only"` flag

  or

- Update the firmware to the latest version available on Mellanox Technologies' Web site as described in section Section 2.4, "Updating Firmware After Installation", on page 46.

2.6 Uninstalling Mellanox OFED

Use the script `/usr/sbin/ofed_uninstall.sh` to uninstall the Mellanox OFED package. The script is part of the `ofed-scripts` RPM.

2.7 Uninstalling Mellanox OFED using the YUM Tool

If MLNX_OFED was installed using the yum tool, then it can be uninstalled as follow:

```
yum groupremove '<group name>'[1]
```

1. The "`<group name>`" must be the same group name that was previously used to install MLNX_OFED.

2.8 UEFI Secure Boot

All kernel modules included in MLNX_OFED for RHEL7 and SLES12 are signed with x.509 key to support loading the modules when Secure Boot is enabled.

2.8.1 Enrolling Mellanox's x.509 Public Key On your Systems

In order to support loading MLNX_OFED drivers when an OS supporting Secure Boot boots on a UEFI-based system with Secure Boot enabled, the Mellanox x.509 public key should be added to the UEFI Secure Boot key database and loaded onto the system key ring by the kernel.

Follow these steps below to add the Mellanox's x.509 public key to your system:

Prior to adding the Mellanox's x.509 public key to your system, please make sure the 'mokutil' package is installed on your system.

Step 1. Download the x.509 public key.

```
# wget http://www.mellanox.com/downloads/ofed/mlnx_signing_key_pub.der
```

Step 2. Add the public key to the MOK list using the mokutil utility.

You will be asked to enter and confirm a password for this MOK enrollment request.

```
# mokutil --import mlnx_signing_key_pub.der
```

Step 3. Reboot the system.

The pending MOK key enrollment request will be noticed by `shim.efi` and it will launch `MokManager.efi` to allow you to complete the enrollment from the UEFI console. You will need to enter the password you previously associated with this request and confirm the enrollment. Once done, the public key is added to the MOK list, which is persistent. Once a key is in the MOK list, it will be automatically propagated to the system key ring and subsequent will be booted when the UEFI Secure Boot is enabled.

To see what keys have been added to the system key ring on the current boot, install the 'keyutils' package and run: `#keyctl list %:.system_keyring`

3 Configuration Files

For the complete list of configuration files, please refer to MLNX_OFED_configuration_files.txt at the following location:
docs/readme_and_user_manual/MLNX_OFED_configuration_files.txt

3.1 Persistent Naming for Network Interfaces

To avoid network interface renaming after boot or driver restart use the `"/etc/udev/rules.d/70-persistent-net.rules"` file.

- Example for Ethernet interfaces:

```
 # PCI device 0x15b3:0x1003 (mlx4_core)
 SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?*", ATTR{address}=="00:02:c9:fa:c3:50",
ATTR{dev_id}=="0x0", ATTR{type}=="1", KERNEL=="eth*", NAME="eth1"
 SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?*", ATTR{address}=="00:02:c9:fa:c3:51",
ATTR{dev_id}=="0x0", ATTR{type}=="1", KERNEL=="eth*", NAME="eth2"
 SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?*", ATTR{address}=="00:02:c9:e9:56:a1",
ATTR{dev_id}=="0x0", ATTR{type}=="1", KERNEL=="eth*", NAME="eth3"
 SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?*", ATTR{address}=="00:02:c9:e9:56:a2",
ATTR{dev_id}=="0x0", ATTR{type}=="1", KERNEL=="eth*", NAME="eth4"
```

- Example for IPoIB interfaces:

```
 SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?*", ATTR{dev_id}=="0x0", ATTR{type}=="32",
NAME="ib0"
 SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?*", ATTR{dev_id}=="0x1", ATTR{type}=="32",
NAME="ib1"
```

4 Driver Features

4.1 SCSI RDMA Protocol

4.1.1 Overview

As described in Section 1.3.4, the SCSI RDMA Protocol (SRP) is designed to take full advantage of the protocol off-load and RDMA features provided by the InfiniBand architecture. SRP allows a large body of SCSI software to be readily used on InfiniBand architecture. The SRP Initiator controls the connection to an SRP Target in order to provide access to remote storage devices across an InfiniBand fabric. The kSRP Target resides in an IO unit and provides storage services.

Section 4.1.2 describes the SRP Initiator included in Mellanox OFED for Linux. This package, however, does *not* include an SRP Target.

4.1.2 SRP Initiator

This SRP Initiator is based on open source from OpenFabrics (www.openfabrics.org) that implements the SCSI RDMA Protocol-2 (SRP-2). SRP-2 is described in Document # T10/1524-D available from http://www.t10.org.

The SRP Initiator supports

- Basic SCSI Primary Commands -3 (SPC-3)
  (www.t10.org/ftp/t10/drafts/spc3/spc3r21b.pdf)
- Basic SCSI Block Commands -2 (SBC-2)
  (www.t10.org/ftp/t10/drafts/sbc2/sbc2r16.pdf)
- Basic functionality, task management and limited error handling

4.1.2.1 Loading SRP Initiator

To load the SRP module, either execute the "`modprobe ib_srp`" command after the OFED driver is up, or change the value of SRP_LOAD in `/etc/infiniband/openib.conf` to "yes".

For the changes to take effect, run: /etc/init.d/openibd restart

When loading the ib_srp module, it is possible to set the module parameter srp_sg_tablesize. This is the maximum number of gather/scatter entries per I/O (default: 12).

4.1.2.1.1 SRP Module Parameters

When loading the SRP module, the following parameters can be set (viewable by the "`modinfo ib_srp`" command):

cmd_sg_entries	Default number of gather/scatter entries in the SRP command (default is 12, max 255)

allow_ext_sg	Default behavior when there are more than cmd_sg_entries S/G entries after mapping; fails the request when false (default false)
topspin_workarounds	Enable workarounds for Topspin/Cisco SRP target bugs
reconnect_delay	Time between successive reconnect attempts. Time between successive reconnect attempts of SRP initiator to a disconnected target until dev_loss_tmo timer expires (if enabled), after that the SCSI target will be removed.
fast_io_fail_tmo	Number of seconds between the observation of a transport layer error and failing all I/O. Increasing this timeout allows more tolerance to transport errors, however, doing so increases the total failover time in case of serious transport failure. Note: fast_io_fail_tmo value must be smaller than the value of reconnect_delay.
dev_loss_tmo	Maximum number of seconds that the SRP transport should insulate transport layer errors. After this time has been exceeded the SCSI target is removed. Normally it is advised to set this to -1 (disabled) which will never remove the scsi_host. In deployments where different SRP targets are connected and disconnected frequently, it may be required to enable this timeout in order to clean old scsi_hosts representing targets that no longer exists.

Constraints between parameters:

- `dev_loss_tmo, fast_io_fail_tmo, reconnect_delay` cannot be all disabled or negative values.
- `reconnect_delay` must be positive number.
- `fast_io_fail_tmo` must be smaller than SCSI block device timeout.
- `fast_io_fail_tmo` must be smaller than `dev_loss_tmo`.

4.1.2.1.2 SRP Remote Ports Parameters

Several SRP remote ports parameters are modifiable online on existing connection.

- ***To modify dev_loss_tmo to 600 seconds:***

```
echo 600 > /sys/class/srp_remote_ports/port-xxx/dev_loss_tmo
```

- ***To modify fast_io_fail_tmo to 15 seconds:***

```
echo 15 > /sys/class/srp_remote_ports/port-xxx/fast_io_fail_tmo
```

- ***To modify reconnect_delay to 10 seconds:***

```
echo 20 > /sys/class/srp_remote_ports/port-xxx/reconnect_delay
```

4.1.2.2 Manually Establishing an SRP Connection

The following steps describe how to manually load an SRP connection between the Initiator and an SRP Target. Section 4.1.2.4 explains how to do this automatically.

- Make sure that the `ib_srp` module is loaded, the SRP Initiator is reachable by the SRP Target, and that an SM is running.
- To establish a connection with an SRP Target and create an SRP (SCSI) device for that target under `/dev`, use the following command:

```
echo -n id_ext=[GUID value],ioc_guid=[GUID value],dgid=[port GID value],\
pkey=ffff,service_id=[service[0] value] > \
/sys/class/infiniband_srp/srp-mlx[hca number]-[port number]/add_target
```

  See Section 4.1.2.3 for instructions on how the parameters in this `echo` command may be obtained.

  Notes:

  - Execution of the above “echo” command may take some time
  - The SM must be running while the command executes
  - It is possible to include additional parameters in the echo command:
    - max_cmd_per_lun - Default: 62
    - max_sect (short for max_sectors) - sets the request size of a command
    - io_class - Default: 0x100 as in rev 16A of the specification (In rev 10 the default was 0xff00)
    - tl_retry_count - a number in the range 2..7 specifying the IB RC retry count. Default: 2
    - comp_vector, a number in the range 0..n-1 specifying the MSI-X completion vector. Some HCA's allocate multiple (n) MSI-X vectors per HCA port. If the IRQ affinity masks of these interrupts have been configured such that each MSI-X interrupt is handled by a different CPU then the comp_vector parameter can be used to spread the SRP completion workload over multiple CPU's.
    - cmd_sg_entries, a number in the range 1..255 that specifies the maximum number of data buffer descriptors stored in the SRP_CMD information unit itself. With allow_ext_sg=0 the parameter cmd_sg_entries defines the maximum S/G list length for a single SRP_CMD, and commands whose S/G list length exceeds this limit after S/G list collapsing will fail.
    - initiator_ext - Please refer to Section 9 (Multiple Connections...)
- To list the new SCSI devices that have been added by the echo command, you may use either of the following two methods:
  - Execute “fdisk -l”. This command lists all devices; the new devices are included in this listing.
  - Execute “dmesg” or look at `/var/log/messages` to find messages with the names of the new devices.

4.1.2.2.1 SRP sysfs Parameters

Interface for making ib_srp connect to a new target. One can request ib_srp to connect to a new target by writing a comma-separated list of login parameters to this sysfs attribute. The supported parameters are:

id_ext	A 16-digit hexadecimal number specifying the eight byte identifier extension in the 16-byte SRP target port identifier. The target port identifier is sent by ib_srp to the target in the SRP_LOGIN_REQ request.

ioc_guid	A 16-digit hexadecimal number specifying the eight byte I/O controller GUID portion of the 16-byte target port identifier.
dgid	A 32-digit hexadecimal number specifying the destination GID.
pkey	A four-digit hexadecimal number specifying the InfiniBand partition key.
service_id	A 16-digit hexadecimal number specifying the InfiniBand service ID used to establish communication with the SRP target. How to find out the value of the service ID is specified in the documentation of the SRP target.
max_sect	A decimal number specifying the maximum number of 512-byte sectors to be transferred via a single SCSI command.
max_cmd_per_lun	A decimal number specifying the maximum number of outstanding commands for a single LUN.
io_class	A hexadecimal number specifying the SRP I/O class. Must be either 0xff00 (rev 10) or 0x0100 (rev 16a). The I/O class defines the format of the SRP initiator and target port identifiers.
initiator_ext	A 16-digit hexadecimal number specifying the identifier extension portion of the SRP initiator port identifier. This data is sent by the initiator to the target in the SRP_LOGIN_REQ request.
cmd_sg_entries	A number in the range 1..255 that specifies the maximum number of data buffer descriptors stored in the SRP_CMD information unit itself. With allow_ext_sg=0 the parameter cmd_sg_entries defines the maximum S/G list length for a single SRP_CMD, and commands whose S/G list length exceeds this limit after S/G list collapsing will fail.
allow_ext_sg	Whether ib_srp is allowed to include a partial memory descriptor list in an SRP_CMD instead of the entire list. If a partial memory descriptor list has been included in an SRP_CMD the remaining memory descriptors are communicated from initiator to target via an additional RDMA transfer. Setting allow_ext_sg to 1 increases the maximum amount of data that can be transferred between initiator and target via a single SCSI command. Since not all SRP target implementations support partial memory descriptor lists the default value for this option is 0.
sg_tablesize	A number in the range 1..2048 specifying the maximum S/G list length the SCSI layer is allowed to pass to ib_srp. Specifying a value that exceeds cmd_sg_entries is only safe with partial memory descriptor list support enabled (allow_ext_sg=1).
comp_vector	A number in the range 0..n-1 specifying the MSI-X completion vector. Some HCA's allocate multiple (n) MSI-X vectors per HCA port. If the IRQ affinity masks of these interrupts have been configured such that each MSI-X interrupt is handled by a different CPU then the comp_vector parameter can be used to spread the SRP completion workload over multiple CPU's.

tl_retry_count	A number in the range 2..7 specifying the IB RC retry count.

4.1.2.3 SRP Tools - ibsrpdm, srp_daemon and srpd Service Script

To assist in performing the steps in Section 6, the OFED distribution provides two utilities, ibsrpdm and srp_daemon, which

- Detect targets on the fabric reachable by the Initiator (for Step 1)
- Output target attributes in a format suitable for use in the above “echo” command (Step 2)
- A service script srpd which may be started at stack startup

The utilities can be found under `/usr/sbin/`, and are part of the srptools RPM that may be installed using the Mellanox OFED installation. Detailed information regarding the various options for these utilities are provided by their man pages.

Below, several usage scenarios for these utilities are presented.

ibsrpdm

ibsrpdm is using for the following tasks:

1. Detecting reachable targets
   a. To detect all targets reachable by the SRP initiator via the default umad device (/sys/class/infiniband_mad/umad0), execute the following command:

      ```
      ibsrpdm
      ```

      This command will output information on each SRP Target detected, in human-readable form.

      Sample output:

      ```
      IO Unit Info:
           port LID:        0103
           port GID:        fe80000000000000002c90200402bd5
           change ID:       0002
           max controllers: 0x10
        controller[  1]
            GUID:      0002c90200402bd4
            vendor ID: 0002c9
            device ID: 005a44
            IO class : 0100
            ID:        LSI Storage Systems SRP Driver 200400a0b81146a1
            service entries: 1
            service[  0]: 200400a0b81146a1 / SRP.T10:200400A0B81146A1
      ```

   b. To detect all the SRP Targets reachable by the SRP Initiator via another umad device, use the following command:

      ```
      ibsrpdm -d <umad device>
      ```

2. Assistance in creating an SRP connection

a. To generate output suitable for utilization in the “echo” command of Section 4.1.2.2, add the ‘-c’ option to ibsrpdm:

```
ibsrpdm -c
```

Sample output:

```
id_ext=200400A0B81146A1,ioc_guid=0002c90200402bd4,
dgid=fe8000000000000002c90200402bd5,pkey=ffff,service_id=200400a0b81146a1
```

b. To establish a connection with an SRP Target using the output from the ‘`ibsrpdm -c`’ example above, execute the following command:

```
echo -n id_ext=200400A0B81146A1,ioc_guid=0002c90200402bd4,
dgid=fe8000000000000002c90200402bd5,pkey=ffff,service_id=200400a0b81146a1 > /sys/
class/infiniband_srp/srp-mlx4_0-1/add_target
```

The SRP connection should now be up; the newly created SCSI devices should appear in the listing obtained from the ‘`fdisk -l`’ command.

3. Discover reachable SRP Targets given an InfiniBand HCA name and port, rather than by just runing `/sys/class/infiniband_mad/umad<N>` where <N> is a digit

srpd

The srpd service script allows automatic activation and termination of the srp_daemon utility on all system live InfiniBand ports.

srp_daemon

The srp_daemon utility is based on ibsrpdm and extends its functionality. In addition to the ibsrpdm functionality described above, srp_daemon can also

- Establish an SRP connection by itself (without the need to issue the “echo” command described in Section 4.1.2.2)
- Continue running in background, detecting new targets and establishing SRP connections with them (daemon mode)
- Discover reachable SRP Targets given an infiniband HCA name and port, rather than just by
  /dev/umad<N> where <N> is a digit
- Enable High Availability operation (together with Device-Mapper Multipath)
- Have a configuration file that determines the targets to connect to

1. `srp_daemon` commands equivalent to ibsrpdm:

```
"srp_daemon -a -o"    is equivalent to "ibsrpdm"
"srp_daemon -c -a -o" is equivalent to "ibsrpdm -c"
```

These srp_daemon commands can behave differently than the equivalent ibsrpdm command when /etc/srp_daemon.conf is not empty.

2. `srp_daemon` extensions to `ibsrpdm`

- To discover SRP Targets reachable from the HCA device <InfiniBand HCA name> and the port <port num>, (and to generate output suitable for 'echo',) you may execute:

```
host1# srp_daemon -c -a -o -i <InfiniBand HCA name> -p <port number>
```

To obtain the list of InfiniBand HCA device names, you can either use the ibstat tool or run ‘ls /sys/class/infiniband’.

- To both discover the SRP Targets and establish connections with them, just add the -e option to the above command.
- Executing srp_daemon over a port without the -a option will only display the reachable targets via the port and to which the initiator is not connected. If executing with the -e option it is better to omit -a.
- It is recommended to use the -n option. This option adds the initiator_ext to the connecting string. (See Section 4.1.2.5 for more details).
- srp_daemon has a configuration file that can be set, where the default is /etc/srp_daemon.conf. Use the -f to supply a different configuration file that configures the targets srp_daemon is allowed to connect to. The configuration file can also be used to set values for additional parameters (e.g., max_cmd_per_lun, max_sect).
- A continuous background (daemon) operation, providing an automatic ongoing detection and connection capability. See Section 4.1.2.4.

4.1.2.4 Automatic Discovery and Connection to Targets

- Make sure that the ib_srp module is loaded, the SRP Initiator can reach an SRP Target, and that an SM is running.
- To connect to all the existing Targets in the fabric, run “`srp_daemon -e -o`”. This utility will scan the fabric once, connect to every Target it detects, and then exit.

srp_daemon will follow the configuration it finds in /etc/srp_daemon.conf. Thus, it will ignore a target that is disallowed in the configuration file.

- To connect to all the existing Targets in the fabric and to connect to new targets that will join the fabric, execute srp_daemon -e. This utility continues to execute until it is either killed by the user or encounters connection errors (such as no SM in the fabric).
- To execute SRP daemon as a daemon on all the ports:
  - `srp_daemon.sh` (found under /usr/sbin/). `srp_daemon.sh` sends its log to `/var/log/srp_daemon.log`.
  - Start the srpd service script, run `service srpd start`
- It is possible to configure this script to execute automatically when the InfiniBand driver starts by changing the value of SRP_DAEMON_ENABLE in /etc/infiniband/openib.conf to “yes”. However, this option also enables SRP High Availability that has some more features – see Section 4.1.2.6).

  For the changes in `openib.conf` to take effect, run:
  `/etc/init.d/openibd restart`

4.1.2.5 Multiple Connections from Initiator InfiniBand Port to the Target

Some system configurations may need multiple SRP connections from the SRP Initiator to the same SRP Target: to the same Target IB port, or to different IB ports on the same Target HCA.

In case of a single Target IB port, i.e., SRP connections use the same path, the configuration is enabled using a different initiator_ext value for each SRP connection. The initiator_ext value is a 16-hexadecimal-digit value specified in the connection command.

Also in case of two physical connections (i.e., network paths) from a single initiator IB port to two different IB ports on the same Target HCA, there is need for a different initiator_ext value on each path. The conventions is to use the Target port GUID as the initiator_ext value for the relevant path.

If you use srp_daemon with -n flag, it automatically assigns initiator_ext values according to this convention. For example:

```
id_ext=200500A0B81146A1,ioc_guid=0002c90200402bec,\
dgid=fe8000000000000002c90200402bed,pkey=ffff,\
service_id=200500a0b81146a1,initiator_ext=ed2b400002c90200
```

Notes:

1. It is recommended to use the -n flag for all srp_daemon invocations.
2. ibsrpdm does not have a corresponding option.
3. srp_daemon.sh always uses the -n option (whether invoked manually by the user, or automatically at startup by setting SRP_DAEMON_ENABLE to yes).

4.1.2.6 High Availability (HA)

Overview

High Availability works using the Device-Mapper (DM) multipath and the SRP daemon. Each initiator is connected to the same target from several ports/HCAs. The DM multipath is responsible for joining together different paths to the same target and for fail-over between paths when one of them goes offline. Multipath will be executed on newly joined SCSI devices.

Each initiator should execute several instances of the SRP daemon, one for each port. At startup, each SRP daemon detects the SRP Targets in the fabric and sends requests to the ib_srp module to connect to each of them. These SRP daemons also detect targets that subsequently join the fabric, and send the ib_srp module requests to connect to them as well.

Operation

When a path (from port1) to a target fails, the ib_srp module starts an error recovery process. If this process gets to the reset_host stage and there is no path to the target from this port, ib_srp will remove this scsi_host. After the scsi_host is removed, multipath switches to another path to this target (from another port/HCA).

When the failed path recovers, it will be detected by the SRP daemon. The SRP daemon will then request ib_srp to connect to this target. Once the connection is up, there will be a new scsi_host for this target. Multipath will be executed on the devices of this host, returning to the original state (prior to the failed path).

Manual Activation of High Availability

Initialization: (Execute after each boot of the driver)

1. Execute modprobe dm-multipath
2. Execute modprobe ib-srp
3. Make sure you have created file /etc/udev/rules.d/91-srp.rules as described above.
4. Execute for each port and each HCA:

   ```
   srp_daemon -c -e -R 300 -i <InfiniBand HCA name> -p <port number>
   ```

   This step can be performed by executing srp_daemon.sh, which sends its log to /var/log/srp_daemon.log.

Now it is possible to access the SRP LUNs on `/dev/mapper/`.

It is possible for regular (non-SRP) LUNs to also be present; the SRP LUNs may be identified by their names. You can configure the /etc/multipath.conf file to change multipath behavior.

It is also possible that the SRP LUNs will not appear under /dev/mapper/. This can occur if the SRP LUNs are in the black-list of multipath. Edit the 'blacklist' section in /etc/multipath.conf and make sure the SRP LUNs are not black-listed.

Automatic Activation of High Availability

- Set the value of SRP_DAEMON_ENABLE in `/etc/infiniband/openib.conf` to "yes".

  For the changes in `openib.conf` to take effect, run:
  `/etc/init.d/openibd restart`
- Start srpd service, run: `service srpd start`
- From the next loading of the driver it will be possible to access the SRP LUNs on /dev/mapper/

It is possible that regular (not SRP) LUNs may also be present; the SRP LUNs may be identified by their name.

- It is possible to see the output of the SRP daemon in /var/log/srp_daemon.log

4.1.2.7 Shutting Down SRP

SRP can be shutdown by using "rmmod ib_srp", or by stopping the OFED driver ("`/etc/init.d/openibd stop`"), or as a by-product of a complete system shutdown.

Prior to shutting down SRP, remove all references to it. The actions you need to take depend on the way SRP was loaded. There are three cases:

1. Without High Availability

   When working without High Availability, you should unmount the SRP partitions that were mounted prior to shutting down SRP.
2. After Manual Activation of High Availability

If you manually activated SRP High Availability, perform the following steps:

a. Unmount all SRP partitions that were mounted.

b. Stop service srpd (Kill the SRP daemon instances).

c. Make sure there are no multipath instances running. If there are multiple instances, wait for them to end or kill them.

d. Run: `multipath -F`

3. After Automatic Activation of High Availability

   If SRP High Availability was automatically activated, SRP shutdown must be part of the driver shutdown ("/etc/init.d/openibd stop") which performs Steps 2-4 of case b above. However, you still have to unmount all SRP partitions that were mounted before driver shutdown.

4.2 iSCSI Extensions for RDMA (iSER)

4.2.1 Overview

iSCSI Extensions for RDMA (iSER) extends the iSCSI protocol to RDMA. It permits data to be transferred directly into and out of SCSI buffers without intermediate data copies.

4.2.2 iSER Initiator

Setting the iSER target is out of scope of this manual. For guidelines of how to do so, please refer to the relevant target documentation (e.g. stgt, clitarget).

The iSER initiator is controlled through the iSCSI interface available from the iscsi-initiator-utils package.

Make sure iSCSI is enabled and properly configured on your system before proceeding with iSER. Additionally, make sure you have RDMA connectivity between the initiator and the target.

```
rping -s [-vVd] [-S size] [-C count] [-a addr] [-p port]
```

Targets settings such as `timeouts` and `retries` are set the same as any other iSCSI targets.

If targets are set to auto connect on boot, and targets are unreachable, it may take a long time to continue the boot process if `timeouts` and `max retries` are set too high.

Example for discovering and connecting targets over iSER:

```
iscsiadm -m discovery -o new -o old -t st -I iser -p <ip:port> -l
```

iSER also supports RoCE without any additional configuration required. To bond the RoCE interfaces, set the `fail_over_mac` option in the bonding driver (see Section 4.4.6, “Bonding IPoIB”, on page 82).

For further information, please refer to Mellanox Community (http://community.mellanox.com/community/solutions/content --> iSER).

4.3 Ethernet over IB (EoIB) vNic

The Ethernet over IB (EoIB) mlx4_vnic module is a network interface implementation over InfiniBand. EoIB encapsulates Layer 2 datagrams over an InfiniBand Datagram (UD) transport service. The InfiniBand UD datagrams encapsulates the entire Ethernet L2 datagram and its payload.

To perform this operation the module performs an address translation from Ethernet layer 2 MAC addresses (48 bits long) to InfiniBand layer 2 addresses made of LID/GID and QPN. This translation is totally invisible to the OS and user. Thus, differentiating EoIB from IPoIB which exposes a 20 Bytes HW address to the OS. The mlx4_vnic module is designed for Mellanox's ConnectX® family of HCAs and intended to be used with Mellanox's BridgeX® gateway family. Having a BridgeX gateway is a requirement for using EoIB. It performs the following operations:

- Enables the layer 2 address translation required by the mlx4_vnic module.
- Enables routing of packets from the InfiniBand fabric to a 1 or 10 GigE Ethernet subnet.

4.3.1 Ethernet over IB Topology

EoIB is designed to work over an InfiniBand fabric and requires the presence of two entities:

- Subnet Manager (SM)

  The required subnet manager configuration is not unique to EoIB but rather similar to other InfiniBand applications and ULPs.

- BridgeX gateway

  The BridgeX gateway is at the heart of EoIB. On one side, usually referred to as the "internal" side, it is connected to the InfiniBand fabric by one or more links. On the other side, usually referred to as the "external" side, it is connected to the Ethernet subnet by one or more ports. The Ethernet connections on the BridgeX's external side are called external ports or eports. Every BridgeX that is in use with EoIB needs to have one or more eports connected.

4.3.1.1 External Ports (eports) and Gateway

The combination of a specific BridgeX box and a specific eport is referred to as a gateway. The gateway is an entity that is visible to the EoIB host driver and is used in the configuration of the network interfaces on the host side. For example, in the host administered vNics the user will request to open an interface on a specific gateway identifying it by the BridgeX box and eport name.

Distinguishing between gateways is essential because they determine the network topology and affect the path that a packet traverses between hosts. A packet that is sent from the host on a specific EoIB interface will be routed to the Ethernet subnet through a specific external port connection on the BridgeX box.

4.3.1.2 Virtual Hubs (vHubs)

Virtual hubs connect zero or more EoIB interfaces (on internal hosts) and an eport through a virtual hub. Each vHub has a unique virtual LAN (VLAN) ID. Virtual hub participants can send packets to one another directly without the assistance of the Ethernet subnet (external side) routing. This means that two EoIB interfaces on the same vHub will communicate solely using the InfiniBand fabric. EoIB interfaces residing on two different vHubs (whether on the same gateway or not) cannot communicate directly.

There are two types of vHubs:

- a default vHub (one per gateway) without a VLAN ID
- vHubs with unique different VLAN IDs

Each vHub belongs to a specific gateway (BridgeX® + eport), and each gateway has one default vHub, and zero or more VLAN-associated vHubs. A specific gateway can have multiple vHubs distinguishable by their unique VLAN ID. Traffic coming from the Ethernet side on a specific eport will be routed to the relevant vHub group based on its VLAN tag (or to the default vHub for that GW if no vLan ID is present).

4.3.1.3 Virtual NIC (vNic)

A virtual NIC is a network interface instance on the host side which belongs to a single vHub on a specific GW. The vNic behaves similar to any regular hardware network interface. The host can have multiple interfaces that belong to the same vHub.

4.3.2 EoIB Configuration

mlx4_vnic module supports two different modes of configuration which is passed to the host mlx4_vnic driver using the EoIB protocol:

- host administration where the vNic is configured on the host side
- network administration where the configuration is done by the BridgeX

Both modes of operation require the presence of a BridgeX gateway in order to work properly. The EoIB driver supports a mixture of host and network administered vNics.

4.3.2.1 EoIB Host Administered vNic

In the host administered mode, vNics are configured using static configuration files located on the host side. These configuration files define the number of vNics, and the vHub that each host administered vNic will belong to (i.e., the vNic's BridgeX box, eport and VLAN id properties). The mlx4_vnic_confd service is used to read these configuration files and pass the relevant data to the mlx4_vnic module. EoIB Host Administered vNic supports two forms of configuration files:

- “Central Configuration File - /etc/infiniband/mlx4_vnic.conf”
- “vNic Specific Configuration Files - ifcfg-ethX”

Both forms of configuration supply the same functionality. If both forms of configuration files exist, the central configuration file has precedence and only this file will be used.

Central Configuration File - /etc/infiniband/mlx4_vnic.conf

The mlx4_vnic.conf file consists of lines, each describing one vNic. The following file format is used:

```
name=eth47 mac=00:25:8B:27:16:84 ib_port=mlx4_0:1 vid=2 vnic_id=7 bx=BX001 eport=A11
```

The fields used in the file have the following meaning:

Table 3 - mlx4_vnic.conf file format

Field	Description
name	The name of the interface that is displayed when running ifconfig.

Table 3 - mlx4_vnic.conf file format

Field	Description
mac	The mac address to assign to the vNic.
ib_port	The device name and port number in the form [device name]:[port number]. The device name can be retrieved by running ibv_devinfo and using the output of hca_id field. The port number can have a value of 1 or 2.
vid	[Optional field] If VLAN ID exists, the vNic will be assigned the specified VLAN ID. This value must be between 0 and 4095. • If the vid is set to 'all', the ALL-VLAN mode will be enabled and the vNic will support multiple vNic tags. • If no vid is specified or value -1 is set, the vNic will be assigned to the default vHub associated with the GW.
vnic_id	A unique number per vNic between 0 and 16K.
bx	The BridgeX box system GUID or system name string.
eport	The string describing the eport name.
pkey	[Optional field] If discovery_pkey module parameter is set, this value will control which partitions would be used to discover the gateways. For more information about discovery_pkeys please refer to Section 4.3.3.5, “Discovery Partitions Configuration”, on page 69

vNic Specific Configuration Files - ifcfg-ethX

EoIB configuration can use the ifcfg-ethX files used by the network service to derive the needed configuration. In such case, a separate file is required per vNic. Additionally, you need to update the ifcfg-ethX file and add some new attributes to it.

On Red Hat Linux, the new file will be of the form:

```
DEVICE=eth2
HWADDR=00:30:48:7d:de:e4
BOOTPROTO=dhcp
ONBOOT=yes
BXADDR=BX001
BXEPORT=A10
VNICIBPORT=mlx4_0:1
VNICVLAN=3  (Optional field)
GW_PKEY=0xfff1
```

The fields used in the file for vNic configuration have the following meaning:

Table 4 - Red Hat Linux mlx4_vnic.conf file format

Field	Description
DEVICE	An optional field. The name of the interface that is displayed when running ifconfig. If it is not present, the trailer of the configuration file name (e.g. ifcfg-eth47 => "eth47") is used instead.

Table 4 - Red Hat Linux mlx4_vnic.conf file format

Field	Description
HWADDR	The mac address to assign the vNic.
BXADDR	The BridgeX box system GUID or system name string.
BXEPORT	The string describing the eport name.
VNICVLAN	[Optional field] If it exists, the vNic will be assigned the VLAN ID specified. This value must be between 0 and 4095 or 'all' for ALL-VLAN feature.
VNICIBPORT	The device name and port number in the form [device name]:[port number]. The device name can be retrieved by running ibv_devinfo and using the output of hca_id field. The port number can have a value of 1 or 2.
GW_PKEY	[Optional field] If discovery_pkey module parameter is set, this value will control on what partition would be used to discover the gateways. For more information about discovery_pkeys please refer to Section 4.3.3.5, "Discovery Partitions Configuration", on page 69

Other fields available for regular eth interfaces in the ifcfg-ethX files may also be used.

mlx4_vnic_confd

Once the configuration files are updated, the host administered vNics can be created. To manage the host administrated vNics, run the following script:

```
Usage: /etc/init.d/mlx4_vnic_confd {start|stop|restart|reload|status}
```

This script manages host administrated vNics only, to retrieve general information about the vNics on the system including network administrated vNics, refer to Section 4.3.3.1, "mlx4_vnic_info", on page 68.

When using BXADDR/bx field, all vNics BX address configuration should be consistent: either all of them use GUID format, or name format.

The MAC and VLAN values are set using the configuration files only, other tools such as (vconfig) for VLAN modification, or (ifconfig) for MAC modification, are not supported.

4.3.2.2 EoIB Network Administered vNic

In network administered mode, the configuration of the vNic is done by the BridgeX®. If a vNic is configured for a specific host, it will appear on that host once a connection is established between the BridgeX and the mlx4_vnic module. This connection between the mlx4_vnic modules and all available BridgeX boxes is established automatically when the mlx4_vnic module is loaded. If the BridgeX is configured to remove the vNic, or if the connection between the host and BridgeX is lost, the vNic interface will disappear (running ifconfig will not display the interface). Similar to host administered vNics, a network administered, vNic resides on a specific vHub.

For further information on how to configure a network administered vNic, please refer to BridgeX documentation.

To disable network administered vNics on the host side load mlx4_vnic module with the net_admin module parameter set to 0.

4.3.2.3 VLAN Configuration

A vNic instance is associated with a specific vHub group. This vHub group is connected to a BridgeX external port and has a VLAN tag attribute. When creating/configuring a vNic you define the VLAN tag it will use via the vid or the VNICVLAN fields (if these fields are absent, the vNic will not have a VLAN tag). The vNic's VLAN tag will be present in all EoIB packets sent by the vNics and will be verified on all packets received on the vNic. When passed from the InfiniBand to Ethernet,the EoIB encapsulation will be disassembled but the VLAN tag will remain.

For example, if the vNic "eth23" is associated with a vHub that uses BridgeX "bridge01", eport "A10" and VLAN tag 8, all incoming and outgoing traffic on eth23 will use a VLAN tag of 8. This will be enforced by both BridgeX and destination hosts. When a packet is passed from the internal fabric to the Ethernet subnet through the BridgeX it will have a "true" Ethernet VLAN tag of 8.

The VLAN implementation used by EoIB uses operating systems unaware of VLANs. This is in many ways similar to switch tagging in which an external Ethernet switch adds/strips tags on traffic preventing the need of OS intervention. EoIB does not support OS aware VLANs in the form of vconfig.

Configuring VLANs

To configure VLAN tag for a vNic, add the VLAN tag property to the configuration file in host administrated mode, or configure the vNic on the appropriate vHub in network administered mode. In the host administered mode when a vHub with the requested VLAN tag is not available, the vNIC's login request will be rejected.

- Host administered VLAN configuration in centralized configuration file can be modified as follow:

  Add "vid=<VLAN tag>" or remove vid property for no VLAN

- Host administered VLAN configuration with ifcfg-ethX configuration files can be modified as follow:

  Add "VNICVLAN=<VLAN tag>" or remove VNICVLAN property for no VLAN

Using a VLAN tag value of 0 is not recommended because the traffic using it would not be separated from non VLAN traffic.

For Host administered vNics, VLAN entry must be set in the BridgeX first. For further information, please refer to BridgeX® documentation.

4.3.2.4 EoIB Multicast Configuration

Configuring Multicast for EoIB interfaces is identical to multicast configuration for native Ethernet interfaces.

EoIB maps Ethernet multicast addresses to InfiniBand MGIDs (Multicast GID). It ensures that different vHubs use mutually exclusive MGIDs. Thus preventing vNics on different vHubs from communicating with one another.

4.3.2.5 EoIB and Quality of Service

EoIB enables the use of InfiniBand service levels. The configuration of the SL is performed through the BridgeX and lets you set different data/control service level values per BridgeX® box.

For further information on the use of non default service levels, please refer to BridgeX documentation.

4.3.2.6 IP Configuration Based on DHCP

Setting an EoIB interface configuration based on DHCP (v3.1.2 which is available via www.isc.org) is performed similarly to the configuration of Ethernet interfaces. When setting the EoIB configuration files, verify that it includes following lines:

- For RedHat: BOOTPROTO=dhcp
- For SLES: BOOTPROTO='dchp'

If EoIB configuration files are included, ifcfg-eth<n> files will be installed under /etc/sysconfig/network-scripts/ on a RedHat machine and under /etc/sysconfig/network/ on a SuSE machine.

DHCP Server

Using a DHCP server with EoIB does not require special configuration. The DHCP server can run on a server located on the Ethernet side (using any Ethernet hardware) or on a server located on the InfiniBand side and running EoIB module.

4.3.2.7 Static EoIB Configuration

To configure a static EoIB you can use an EoIB configuration that is not based on DHCP. Static configuration is similar to a typical Ethernet device configuration. For further information on how to configure IP addresses, please refer to your Linux distribution documentation.

Ethernet configuration files are located at /etc/sysconfig/network-scripts/ on a RedHat machine and at /etc/sysconfig/network/ on a SuSE machine.

4.3.2.8 Sub Interfaces (VLAN)

EoIB interfaces do not support creating sub interfaces via the vconfig command, unless working in ALL VLAN mode.. To create interfaces with VLAN, refer to Section , “Configuring VLANs”, on page 66.

4.3.3 Retrieving EoIB Information

4.3.3.1 mlx4_vnic_info

To retrieve information regarding EoIB interfaces, use the script mlx4_vnic_info. This script provides detailed information about a specific vNic or all EoIB vNic interfaces, such as: BX info, IOA info, SL, PKEY, Link state and interface features. If network administered vNics are enabled, this script can also be used to discover the available BridgeX® boxes from the host side.

- To discover the available gateway, run:

```
mlx4_vnic_info -g
```

- To receive the full vNic information of eth10, run:

```
mlx4_vnic_info -i eth10
```

- To receive a shorter information report on eth10, run:

```
mlx4_vnic_info -s eth10
```

- To get help and usage information, run:

```
mlx4_vnic_info --help
```

4.3.3.2 Link State

An EoIB interface can report two different link states:

- The physical link state of the interface that is made up of the actual HCA port link state and the status of the vNics connection with the BridgeX®. If the HCA port link state is down or the EoIB connection with the BridgeX has failed, the link will be reported as down because without the connection to the BridgeX the EoIB protocol cannot work and no data can be sent on the wire. The mlx4_vnic driver can also report the status of the external BridgeX port status by using the mlx4_vnic_info script. If the eport_state_enforce module parameter is set, then the external port state will be reported as the vNic interface link state. If the connection between the vNic and the BridgeX is broken (hence the external port state is unknown)the link will be reported as down.
- the link state of the external port associated with the vNic interface

A link state is down on a host administrated vNic, when the BridgeX is connected and the InfiniBand fabric appears to be functional. The issue might result from a misconfiguration of either BXADDR or/and BXEPORT configuration file.

To query the link state run the following command and look for "Link detected":

```
ethtool <interface name>
```

Example:

```
ethtool eth10
Settings for eth10:
Supported ports: [ ]
Supported link modes:
Supports auto-negotiation: No
```

```
Advertised link modes:  Not reported
Advertised auto-negotiation: No
Speed: Unknown! (10000)
Duplex: Full
Port: Twisted Pair
PHYAD: 0
Transceiver: internal
Auto-negotiation: off
Supports Wake-on: d
Wake-on: d
Current message level: 0x00000000 (0)
Link detected: yes
```

4.3.3.3 Bonding Driver

EoIB uses the standard Linux bonding driver. For more information on the Linux Bonding driver please refer to:

<kernel-source>/Documentation/networking/bonding.txt.

Currently only fail-over modes are supported by the EoIB driver, load-balancing modes including static and dynamic (LACP) configurations are not supported.

4.3.3.4 Jumbo Frames

EoIB supports jumbo frames up to the InfiniBand limit of 4K bytes. The default Maximum Transmit Unit (MTU) for EoIB driver is 1500 bytes.

➢ ***To configure EoIB to work with jumbo frames:***

1. Make sure that the IB HCA and Switches hardware support 4K MTU.
2. Configure Mellanox low level driver to support 4K MTU. Add:

```
mlx4_core module parameter to set_4k_mtu=1
```

3. Change the MTU value of the vNic, for example, run:

```
ifconfig eth2 mtu 4038
```

Due to EoIB protocol overhead, the maximum MTU value that can be set for the vNic interface is: 4038 bytes. If the vNic is configured to use VLANs, then the maximum MTU is: 4034 bytes (due to VLAN header insertion).

4.3.3.5 Discovery Partitions Configuration

EoIB enables mapping of VLANs to InfiniBand partitions. Mapping VLANs to partitions causes all EoIB data traffic and all vNic related control traffic to be sent to the mapped partitions. In rare cases, it might be useful to ensure that EoIB discovery packets (packets used for discovery of Gateways (GWs) and vice versa) are sent to a non default partition. This might be used to limit and enforce the visibility of GWs by different hosts.

The **`discovery_pkeys`** module parameter can be used to define which partitions would be used to discove the GWs. The module parameters allow the using of up to 24 different PKEYs. If not set, the default PKEY will be used, and only GWs using the default PKEY would be discovered.

For example, to configure a host to discover GWs on three partitions 0xffff,0xfff1 and 0x3 add the following line to modprobe configuration file:

```
options mlx4_vnic discovery_pkeys=0xffff,0xfff1,0x3
```

When using this feature combined with host administrated vnics, each vnic should also be configured with the partition it should be created on.

For example, for creating host admin vnic on I/F eth20, with pkey 0xfff1 add the following line to ifcg-eth20:

```
GW_PKEY=0xfff1
```

When using a non default partition, the GW partitions should also be configured on the GW in the BridgeX. Additionally, the Subnet Manager must be configured accordingly.

4.3.3.6 ALL VLAN

In Ethernet over InfiniBand (EoIB), a vNic is a member of a vHUB that uniquely defines its Virtual Local Area Networks (VLAN) tag. The VLAN tag is used in the VLAN header within the EoIB packets, and is enforced by EoIB hosts when handling the EoIB packets. The tag is also extended to the Ethernet fabric when packets pass through the BridgeX®. This model of operation ensures a high level of security however, it requires each VLAN tag used to have its own individual vNic to be created and each vHub requires InfiniBand fabric resources like multicast groups (MGIDs).

If many VLANs are needed, the resources required to create and manage them are large. ALL VLAN vHub enables the user to use its resources efficiently by creating a vNic that can support multiple VLAN tags without creating multiple vNics. However, it reduces VLAN separation compared to the vNic /vHub model.

ALL VLAN Functionality

When ALL VLAN is enabled, the address lookup on the BridgeX® consists of the MAC address only (without the VLAN), so all packets with the same MAC regardless of the VLAN, are sent to the same InfiniBand address. Same behavior can be expected from the host EoIB driver, which also sends packets to the relevant InfiniBand addresses while disregarding the VLAN. In both scenarious, the Ethernet packet that is embedded in the EoIB packet includes the VLAN header enabling VLAN enforcement either in the Ethernet fabric or at the receiving EoIB host.

ALL VLAN must be supported by both the BridgeX® and by the host side.

When enabling ALL VLAN, all gateways (LAG or legacy) that have eports belonging to a gateway group (GWG) must be configured to the same behavior.

For example it is impossible to have gateway A2 configured to all-vlan mode and A3 to regular mode, because both belong to GWG A.

A gateway that is configured to work in ALL VLAN mode cannot accept login requests from

- vNics that do not support this mode
- host admin vNics that were not configured to work in ALL VLAN mode, by setting the vlan-id value to a 'all' as as described in Section , “Creating vNICs that Support ALL VLAN Mode”, on page 71.

Creating vNICs that Support ALL VLAN Mode

VLANs are created on a vNIC that supports ALL VLAN mode using "vconfig".

- **net-admin vNics**

  The net-admin vNic supports ALL VLAN mode once it is created on a gateway configured with All-VLAN mode.

- **host-admin vNics**

  - To create an ALL VLAN vnic, set the VLAN's ID to 'all'.

    A gateway that is configured to work in ALL VLAN mode, can only accept login requests from hosts that are also working in a VLAN mode. e.g. the VLAN ID must be set to 'all'.

  This is an example of how to create an ALL VLAN vNic using the mlx4_vnic.conf file:

```
 name=eth44 mac=00:25:8B:27:14:78 ib_port=mlx4_0:1 vid=all vnic_id=5
bx=00:00:00:00:00:00:04:B2 eport=A10
```

  - To create an All-VLAN vNic using a specific configuration file, add the following line to the configuration file:

```
VNICVLAN=all
```

For further information on how to create host-admin vNics, please see Section 4.3.2.1, “EoIB Host Administered vNic”, on page 63 EoIB Host Administered vNic.

Checking the Configuration

To verify the gateway / vNic is configured with the All-VLAN mode, use the mlx4_vnic_info script.

- **Gateway Support**

  To verify the gateway is configured to All-VLAN mode. Run:

```
mlx4_vnic_info -g <GW-NAME>
```

  Example:

```
# mlx4_vnic_info -g A2
IOA_PORT      mlx4_0:1
BX_NAME       bridge-119c64
BX_GUID       00:02:c9:03:00:11:61:67
EPORT_NAME    A2
EPORT_ID      63
STATE         connected
GW_TYPE       LEGACY
PKEY          0xffff
ALL_VLAN      yes
```

- **vNic Support**

  To verify the vNIC is configured to All-VLAN mode. Run:

  ```
  mlx4_vnic_info -i <interface>
  ```

  Example:

  ```
  # mlx4_vnic_info -i eth204
  NETDEV_NAME   eth204
  NETDEV_LINK   up
  NETDEV_OPEN   yes
  GW_TYPE       LEGACY
  ALL_VLAN      yes
  ```

For further information on mlx4_vnic_info script, please see Section 4.3.3.1, “mlx4_vnic_info”, on page 68.

4.3.4 Advanced EoIB Settings

4.3.4.1 Module Parameters

The mlx4_vnic driver supports the following module parameters. These parameters are intended to enable more specific configuration of the mlx4_vnic driver to customer needs. The mlx4_vnic is also effected by module parameters of other modules such as set_4k_mtu of mlx4_core. This modules are not addressed in this section.

The available module parameters include:

- tx_rings_num: Number of TX rings, use 0 for #cpus [default 0, max 32]
- tx_rings_len: Length of TX rings, must be power of two [default 1024, max 8K]
- rx_rings_num: Number of RX rings, use 0 for #cpus [default 0, max 32]
- rx_rings_len: Length of RX rings, must be power of two [default 2048, max 8K]
- vnic_net_admin: Network administration enabled [default 1]
- lro_num: Number of LRO sessions per ring, use 0 to disable LRO [default 32, max 32]
- eport_state_enforce: Bring vNic up only when corresponding External Port is up [default 0]
- discovery_pkeys: Vector of up to 24 PKEYs to be used for discovery [default 0xFFFF] (array of int)

For all module parameters list and description, run:

```
mlx4_vnic_info -I
```

To check the current module parameters, run:

```
mlx4_vnic_info -P
```

Default RX/TX rings number is the number of logical CPUs (threads). To set non-default values to module parameters, the following line should be added to modprobe configuration file (e.g. /etc/modprobe.conf file):

```
options mlx4_vnic <param_name>=<value> <param_name>=<value> ...
```

For additional information about **`discovery_pkeys`** please refer to Section 4.3.3.5, “Discovery Partitions Configuration”, on page 69

4.3.4.2 vNic Interface Naming

The mlx4_vnic driver enables the kernel to determine the name of the registered vNic. By default, the Linux kernel assigns each vNic interface the name eth<N>, where <N> is an incremental number that keeps the interface name unique in the system. The vNic interface name may not remain consistent among hosts or BridgeX reboots as the vNic creation can happen in a different order each time. Therefore, the interface name may change because of a "first-come-first-served" kernel policy. In automatic network administered mode, the vNic MAC address may also change, which makes it difficult to keep the interface configuration persistent.

To control the interface name, you can use standard Linux utilities such as IFRENAME(8), IP(8) or UDEV(7). For example, to change the interface eth2 name to eth.bx01.a10, run:

```
ifrename -i eth2 -n eth.bx01.a10
```

To generate a unique vNic interface name, use the mlx4_vnic_info script with the '-u' flag. The script will generate a new name based on the scheme:

```
eth<pci-id>.<ib-port-num>.<gw_port_id>.[vlan-id]
```

For example, if vNic eth2 resides on an InfiniBand card on the PCI BUS ID 0a:00.0 PORT #1, and is connected to the GW PORT ID #3 without VLAN, its unique name will be:

```
mlx4_vnic_info -u eth2
eth2   eth10.1.3
```

You can add your own custom udev rule to use the output of the script and to rename the vNic interfaces automatically. To create a new udev rule file under /etc/udev/rules.d/61-vnic-net.rules, include the line:

```
SUBSYSTEM=="net", PROGRAM=="/sbin/mlx4_vnic_info -u %k", NAME="%c{2+}"
```

UDEV service is active by default however if it is not active, run:

```
/sbin/udevd -d
```

When vNic MAC address is consistent, you can statically name each interface using the UDEV following rule:

```
SUBSYSTEM=="net", SYSFS{address}=="aa:bb:cc:dd:ee:ff", NAME="ethX"
```

For further information on the UDEV rules syntax, please refer to udev man pages.

4.3.4.3 Para-Virtualized vNic

EoIB driver interfaces can be also used for Linux based virtualization environment such as Xen/KVM based Hypervisors. This section explains how to configure Para-Virtualized (PV) EoIB to work in such an environment.

Driver Configuration

For PV-EoIB to work properly, the following features must be disabled in the driver:

- Large Receive Offload (LRO)
- TX completion polling
- RX fragmented buffers

To disable the features above, edit the modprobe configuration file as follow:

```
options mlx4_vnic lro_num=0 tx_polling=0 rx_linear=1
```

For the full list of mlx4_vnic module parameters, run:

```
# modinfo mlx4_vnic
```

Network Configuration

PV-EoIB supports both L2 (bridged) and L3 (routed) network models. The 'physical' interfaces that can be enslaved to the Hypervisor virtual bridge are actually EoIB vNics, and they can be created as on an native Linux machine. PV-EoIB driver supports both host-administrated and network-administrated vNics. Please refer to Section 4.3.2, “EoIB Configuration”, on page 63 for more information on vNics configuration.

Once an EoIB vNic is enslaved to a virtual bridge, it can be used by any Guest OS that is supported by the Hypervisor. The driver will automatically manage the resources required to serve the Guest OS network virtual interfaces (based on their MAC address).

To see the list of MAC addresses served by an EoIB vNic, log into the Hypervisor and run the command:

```
# mlx4_vnic_info -m <interface>
```

The driver detects virtual interfaces MAC addresses based in their outgoing packets, so you may notice that the virtual MAC address is being detected by the EoIB driver only after the first packet is sent out by the Guest OS. Virtual resources MAC addresses cleanup is managed by mlx4_vnic daemon as explained in Section , “Resources Cleanup”, on page 75.

Multicast Configuration

Virtual machines multicast traffic over PV-EoIB is supported in promiscuous mode. Hence, all multicast traffic is sent over the broadcast domain, and filtered in the VM level.

- To enable promiscuous multicast, log into the BridgeX CLI and run the command:

  ```
  # bxm eoib mcast promiscuous
  ```

  Please refer to BridgeX CLI Guide for additional details.

- To see the multicast configuration of a vNic from the host, log into the Hypervisor and run:

  ```
  # mlx4_vnic_info -i <interface> | grep MCAST
  ```

VLANs

Virtual LANs are supported in EoIB vNic level, where VLAN tagging/untagging is done by the EoIB driver.

- To enable VLANs on top of a EoIB vNic:
  a. Create a new vNic interface with the corresponding VLAN ID
  b. Enslave it to a virtual bridge to be used by the Guest OS. The VLAN tagging/untagging is transparent to the Guest and managed in EoIB driver level.

The vconfig utility is not supported by EoIB driver, a new vNic instance must be created instead. For further information, see Section 4.3.2.3, “VLAN Configuration”, on page 66.

Virtual Guest Tagging (VGT) is not supported. The model explained above applies to Virtual Switch Tagging (VST) only.

Migration

Some Hypervisors provide the ability to migrate a virtual machine from one physical server to another, this feature is seamlessly supported by PV-EoIB. Any network connectivity over EoIB will automatically be resumed on the new physical server. The downtime that may occur during this process is minor.

Resources Cleanup

When a virtual interface within the Guest OS is no longer connected to an EoIB link, its MAC address need to be removed from the EoIB driver. The cleaning is managed by the Garbage Collector (GC) service. The GC functionality is included in the mlx4_vnic daemon (python script):

```
# /sbin/mlx4_vnicd
```

- To enable/disable the mlx4_vnic daemon,
  a. Edit the **/etc/infiniband/mlx4_vnic.conf** file by including the line:

```
# mlx4_vnicd=<yes|no> [parameters]
```

  b. Start the service **mlx4_vnic_confd** to read and apply the configuration:

```
# /etc/init.d/mlx4_vnic_confd start
```

- To see full list of the daemon parameters, run:

```
# mlx4_vnicd --help
```

For example, to enable mlx4_vnic daemon with GC:

```
# cat /etc/infiniband/mlx4_vnic.conf
  mlx4_vnicd=yes gc_enable=yes
# /etc/init.d/mlx4_vnic_confd start
  Checking configuration file:                          [  OK  ]
  Starting mlx4_vnicd (pid 30920):                      [  OK  ]
```

The mlx4_vnicd daemon requires xenstore or libvirt to run.

Some Hypervisors may not have enough memory for the driver domain, as a result **mlx4_vnic** driver may fail to initialize or create more vNics, causing the machine to be unresponsive.

- To avoid this behavior, you can:
  a. Allocate more memory for the driver domain.

     For further information on how to increase dom0_mem, please refer to:

     http://support.citrix.com/article/CTX126531

  b. Lower the **mlx4_vnic** driver memory consumption by decreasing its RX/TX rings number and length,

     For further information, please refer to Section 4.3.4.1, “Module Parameters”, on page 72.

4.4 IP over InfiniBand

4.4.1 Introduction

The IP over IB (IPoIB) driver is a network interface implementation over InfiniBand. IPoIB encapsulates IP datagrams over an InfiniBand Connected or Datagram transport service. The IPoIB driver, ib_ipoib, exploits the following capabilities:

- VLAN simulation over an InfiniBand network via child interfaces
- High Availability via Bonding
- Varies MTU values:
  - up to 4k in Datagram mode
  - up to 64k in Connected mode
- Uses any ConnectX® IB ports (one or two)
- Inserts IP/UDP/TCP checksum on outgoing packets
- Calculates checksum on received packets
- Support net device TSO through ConnectX® LSO capability to defragment large datagrams to MTU quantas.
- Dual operation mode - datagram and connected
- Large MTU support through connected mode

IPoIB also supports the following software based enhancements:

- Giant Receive Offload
- NAPI
- Ethtool support

4.4.2 IPoIB Mode Setting

IPoIB can run in two modes of operation: Connected mode and Datagram mode. By default, IPoIB is set to work in Datagram, except for Connect-IB™ adapter card which uses IPoIB with Connected mode as default.

For better scalability and performance we recommend using the Datagram mode. However, the mode can be changed to Connected mode by editing the file `/etc/infiniband/openib.conf` and setting `'SET_IPOIB_CM=yes'`.

The `SET_IPOIB_CM` parameter is set to `"auto"` by default to enable the Connected mode for Connect-IB™ card and Datagram for all other ConnectX® cards.

After changing the mode, you need to restart the driver by running:

```
/etc/init.d/openibd restart
```

To check the current mode used for out-going connections, enter:

```
cat /sys/class/net/ib<n>/mode
```

4.4.3 IPoIB Configuration

Unless you have run the installation script `mlnxofedinstall` with the flag '-n', then IPoIB has not been configured by the installation. The configuration of IPoIB requires assigning an IP address and a subnet mask to each HCA port, like any other network adapter card (i.e., you need to prepare a file called ifcfg-ib<n> for each port). The first port on the first HCA in the host is called interface ib0, the second port is called ib1, and so on.

An IPoIB configuration can be based on DHCP (Section 4.4.3.1) or on a static configuration (Section 4.4.3.2) that you need to supply. You can also apply a manual configuration that persists only until the next reboot or driver restart (Section 4.4.3.3).

4.4.3.1 IPoIB Configuration Based on DHCP

Setting an IPoIB interface configuration based on DHCP is performed similarly to the configuration of Ethernet interfaces. In other words, you need to make sure that IPoIB configuration files include the following line:

For RedHat:

```
BOOTPROTO=dhcp
```

For SLES:

```
BOOTPROTO='dchp'
```

If IPoIB configuration files are included, ifcfg-ib<n> files will be installed under:
/etc/sysconfig/network-scripts/ on a RedHat machine
/etc/sysconfig/network/ on a SuSE machine.

A patch for DHCP may be required for supporting IPoIB. For further information, please see the REAME which is available under the docs/dhcp/ directory.

Standard DHCP fields holding MAC addresses are not large enough to contain an IPoIB hardware address. To overcome this problem, DHCP over InfiniBand messages convey a client identifier field used to identify the DHCP session. This client identifier field can be used to associate an IP address with a client identifier value, such that the DHCP server will grant the same IP address to any client that conveys this client identifier.

The length of the client identifier field is not fixed in the specification. For the *Mellanox OFED for Linux* package, it is recommended to have IPoIB use the same format that FlexBoot uses for this client identifier – see Section A.3.2, "Configuring the DHCP Server", on page 259.

4.4.3.1.1 DHCP Server

In order for the DHCP server to provide configuration records for clients, an appropriate configuration file needs to be created. By default, the DHCP server looks for a configuration file called `dhcpd.conf` under `/etc`. You can either edit this file or create a new one and provide its full path to the DHCP server using the -cf flag (See a file example at `docs/dhcpd.conf`).

The DHCP server must run on a machine which has loaded the IPoIB module.

To run the DHCP server from the command line, enter:

```
dhcpd <IB network interface name> -d
```

Example:

```
host1# dhcpd ib0 -d
```

4.4.3.1.2 DHCP Client (Optional)

A DHCP client can be used if you need to prepare a diskless machine with an IB driver. See Step 8 under "Example: Adding an IB Driver to initrd (Linux)".

In order to use a DHCP client identifier, you need to first create a configuration file that defines the DHCP client identifier.
Then run the DHCP client with this file using the following command:

```
dhclient -cf <client conf file> <IB network interface name>
```

Example of a configuration file for the ConnectX (PCI Device ID 26428), called `dhclient.conf`:

```
# The value indicates a hexadecimal number
interface "ib1" {
send dhcp-client-identifier
ff:00:00:00:00:00:02:00:00:02:c9:00:00:02:c9:03:00:00:10:39;
}
```

Example of a configuration file for InfiniHost III Ex (PCI Device ID 25218), called `dhclient.conf`:

```
# The value indicates a hexadecimal number
interface "ib1" {
send dhcp-client-identifier
20:00:55:04:01:fe:80:00:00:00:00:00:00:00:02:c9:02:00:23:13:92;
}
```

In order to use the configuration file, run:

```
host1# dhclient -cf dhclient.conf ib1
```

4.4.3.2 Static IPoIB Configuration

If you wish to use an IPoIB configuration that is not based on DHCP, you need to supply the installation script with a configuration file (using the '-n' option) containing the full IP configuration. The IPoIB configuration file can specify either or both of the following data for an IPoIB interface:

- A static IPoIB configuration
- An IPoIB configuration based on an Ethernet configuration

  See your Linux distribution documentation for additional information about configuring IP addresses.

The following code lines are an excerpt from a sample IPoIB configuration file:

```
# Static settings; all values provided by this file
IPADDR_ib0=11.4.3.175
NETMASK_ib0=255.255.0.0
NETWORK_ib0=11.4.0.0
BROADCAST_ib0=11.4.255.255
ONBOOT_ib0=1
# Based on eth0; each '*' will be replaced with a corresponding octet
# from eth0.
LAN_INTERFACE_ib0=eth0
IPADDR_ib0=11.4.'*'.'*'
NETMASK_ib0=255.255.0.0
NETWORK_ib0=11.4.0.0
BROADCAST_ib0=11.4.255.255
ONBOOT_ib0=1
# Based on the first eth<n> interface that is found (for n=0,1,...);
# each '*' will be replaced with a corresponding octet from eth<n>.
LAN_INTERFACE_ib0=
IPADDR_ib0=11.4.'*'.'*'
NETMASK_ib0=255.255.0.0
NETWORK_ib0=11.4.0.0
BROADCAST_ib0=11.4.255.255
ONBOOT_ib0=1
```

4.4.3.3 Manually Configuring IPoIB

This manual configuration persists only until the next reboot or driver restart.

To manually configure IPoIB for the default IB partition (VLAN), perform the following steps:

Step 1. To configure the interface, enter the `ifconfig` command with the following items:

- The appropriate IB interface (ib0, ib1, etc.)
- The IP address that you want to assign to the interface
- The netmask keyword

- The subnet mask that you want to assign to the interface

The following example shows how to configure an IB interface:

```
host1$ ifconfig ib0 11.4.3.175 netmask 255.255.0.0
```

Step 2. (Optional) Verify the configuration by entering the `ifconfig` command with the appropriate interface identifier *ib#* argument.

The following example shows how to verify the configuration:

```
host1$ ifconfig ib0
b0  Link encap:UNSPEC  HWaddr 80-00-04-04-FE-80-00-00-00-00-00-00-00-00-00-00
inet addr:11.4.3.175  Bcast:11.4.255.255  Mask:255.255.0.0
UP BROADCAST MULTICAST  MTU:65520  Metric:1
RX packets:0 errors:0 dropped:0 overruns:0 frame:0
TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:128
RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)
```

Step 3. Repeat Step 1 and Step 2 on the remaining interface(s).

4.4.4 Subinterfaces

You can create subinterfaces for a primary IPoIB interface to provide traffic isolation. Each such subinterface (also called a child interface) has a different IP and network addresses from the primary (parent) interface. The default Partition Key (PKey), ff:ff, applies to the primary (parent) interface.

This section describes how to

- Create a subinterface (Section 4.4.4.1)
- Remove a subinterface (Section 4.4.4.2)

4.4.4.1 Creating a Subinterface

In the following procedure, ib0 is used as an example of an IB subinterface.

To create a child interface (subinterface), follow this procedure:

Step 1. Decide on the PKey to be used in the subnet (valid values can be 0 or any 16-bit unsigned value). The actual PKey used is a 16-bit number with the most significant bit set. For example, a value of 1 will give a PKey with the value 0x8001.

Step 2. Create a child interface by running:

```
host1$ echo <PKey> > /sys/class/net/<IB subinterface>/create_child
```

Example:

```
host1$ echo 1 > /sys/class/net/ib0/create_child
```

This will create the interface ib0.8001.

Step 3. Verify the configuration of this interface by running:

```
host1$ ifconfig <subinterface>.<subinterface PKey>
```

Using the example of Step 2:

```
host1$ ifconfig ib0.8001
ib0.8001  Link encap:UNSPEC  HWaddr 80-00-00-4A-FE-80-00-00-00-00-00-00-00-00-00-00
BROADCAST MULTICAST  MTU:2044  Metric:1
RX packets:0 errors:0 dropped:0 overruns:0 frame:0
TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:128
RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)
```

Step 4. As can be seen, the interface does not have IP or network addresses. To configure those, you should follow the manual configuration procedure described in Section 4.4.3.3.

Step 5. To be able to use this interface, a configuration of the Subnet Manager is needed so that the PKey chosen, which defines a broadcast address, be recognized (see Chapter 8, "OpenSM – Subnet Manager").

4.4.4.2 Removing a Subinterface

To remove a child interface (subinterface), run:

```
echo <subinterface PKey> /sys/class/net/<ib_interface>/delete_child
```

Using the example of Step 2:

```
echo 0x8001 > /sys/class/net/ib0/delete_child
```

Note that when deleting the interface you must use the PKey value with the most significant bit set (e.g., 0x8000 in the example above).

4.4.5 Verifying IPoIB Functionality

To verify your configuration and your IPoIB functionality, perform the following steps:

Step 1. Verify the IPoIB functionality by using the `ifconfig` command.

The following example shows how two IB nodes are used to verify IPoIB functionality. In the following example, IB node 1 is at 11.4.3.175, and IB node 2 is at 11.4.3.176:

```
host1# ifconfig ib0 11.4.3.175 netmask 255.255.0.0
host2# ifconfig ib0 11.4.3.176 netmask 255.255.0.0
```

Step 2. Enter the ping command from 11.4.3.175 to 11.4.3.176.

The following example shows how to enter the ping command:

```
host1# ping -c 5 11.4.3.176
PING 11.4.3.176 (11.4.3.176) 56(84) bytes of data.
64 bytes from 11.4.3.176: icmp_seq=0 ttl=64 time=0.079 ms
64 bytes from 11.4.3.176: icmp_seq=1 ttl=64 time=0.044 ms
64 bytes from 11.4.3.176: icmp_seq=2 ttl=64 time=0.055 ms
64 bytes from 11.4.3.176: icmp_seq=3 ttl=64 time=0.049 ms
64 bytes from 11.4.3.176: icmp_seq=4 ttl=64 time=0.065 ms
--- 11.4.3.176 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 3999ms rtt min/avg/max/mdev =
0.044/0.058/0.079/0.014 ms, pipe 2
```

4.4.6 Bonding IPoIB

To create an interface configuration script for the ibX and bondX interfaces, you should use the standard syntax (depending on your OS).

Bonding of IPoIB interfaces is accomplished in the same manner as would bonding of Ethernet interfaces: via the Linux Bonding Driver.

- Network Script files for IPoIB slaves are named after the IPoIB interfaces (e.g: ifcfg-ib0)
- The only meaningful bonding policy in IPoIB is High-Availability (bonding mode number 1, or active-backup)
- Bonding parameter "fail_over_mac" is meaningless in IPoIB interfaces, hence, the only supported value is the default: 0 (or "none" in SLES11)

For a persistent bonding IPoIB Network configuration, use the same Linux Network Scripts semantics, with the following exceptions/ additions:

- In the bonding master configuration file (e.g: ifcfg-bond0), in addition to Linux bonding semantics, use the following parameter: MTU=65520

65520 is a valid MTU value only if all IPoIB slaves operate in Connected mode (See Section 4.4.2, “IPoIB Mode Setting”, on page 76) and are configured with the same value. For IPoIB slaves that work in datagram mode, use MTU=2044. If you do not set the correct MTU or do not set MTU at all, performance of the interface might decrease.

- In the bonding slave configuration file (e.g: ifcfg-ib0), use the same Linux Network Scripts semantics. In particular: DEVICE=ib0
- In the bonding slave configuration file (e.g: ifcfg-ib0.8003), the line TYPE=InfiniBand is necessary when using bonding over devices configured with partitions (p_key)
- For RHEL users:

  In /etc/modprobe.b/bond.conf add the following lines:

  ```
  alias bond0 bonding
  ```

- For SLES users:

  It is necessary to update the MANDATORY_DEVICES environment variable in /etc/sysconfig/network/config with the names of the IPoIB slave devices (e.g. ib0, ib1, etc.). Otherwise, bonding master may be created before IPoIB slave interfaces at boot time.

  It is possible to have multiple IPoIB bonding masters and a mix of IPoIB bonding master and Ethernet bonding master. However, It is NOT possible to mix Ethernet and IPoIB slaves under the same bonding master

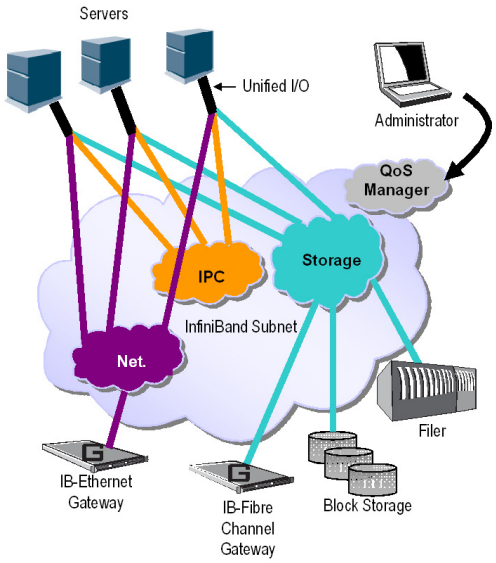
Restarting openibd does no keep the bonding configuration via Network Scripts. You have to restart the network service in order to bring up the bonding master. After the configuration is saved, restart the network service by running: /etc/init.d/network restart.

4.5 Quality of Service InfiniBand

4.5.1 Quality of Service Overview

Quality of Service (QoS) requirements stem from the realization of I/O consolidation over an IB network. As multiple applications and ULPs share the same fabric, a means is needed to control their use of network resources.

Figure 2: I/O Consolidation Over InfiniBand



QoS over Mellanox OFED for Linux is discussed in Chapter 8, “OpenSM – Subnet Manager”.

The basic need is to differentiate the service levels provided to different traffic flows, such that a policy can be enforced and can control each flow utilization of fabric resources.

The InfiniBand Architecture Specification defines several hardware features and management interfaces for supporting QoS:

- Up to 15 Virtual Lanes (VL) carry traffic in a non-blocking manner
- Arbitration between traffic of different VLs is performed by a two-priority-level weighted round robin arbiter. The arbiter is programmable with a sequence of (VL, weight) pairs and a maximal number of high priority credits to be processed before low priority is served
- Packets carry class of service marking in the range 0 to 15 in their header SL field
- Each switch can map the incoming packet by its SL to a particular output VL, based on a programmable table VL=SL-to-VL-MAP(in-port, out-port, SL)
- The Subnet Administrator controls the parameters of each communication flow by providing them as a response to Path Record (PR) or MultiPathRecord (MPR) queries

DiffServ architecture (IETF RFC 2474 & 2475) is widely used in highly dynamic fabrics. The following subsections provide the functional definition of the various software elements that enable a DiffServ-like architecture over the Mellanox OFED software stack.

4.5.2 QoS Architecture

QoS functionality is split between the SM/SA, CMA and the various ULPs. We take the "chronology approach" to describe how the overall system works.

1. The network manager (human) provides a set of rules (policy) that define how the network is being configured and how its resources are split to different QoS-Levels. The policy also define how to decide which QoS-Level each application or ULP or service use.
2. The SM analyzes the provided policy to see if it is realizable and performs the necessary fabric setup. Part of this policy defines the default QoS-Level of each partition. The SA is enhanced to match the requested Source, Destination, QoS-Class, Service-ID, PKey against the policy, so clients (ULPs, programs) can obtain a policy enforced QoS. The SM may also set up partitions with appropriate IPoIB broadcast group. This broadcast group carries its QoS attributes: SL, MTU, RATE, and Packet Lifetime.
3. IPoIB is being setup. IPoIB uses the SL, MTU, RATE and Packet Lifetime available on the multicast group which forms the broadcast group of this partition.
4. MPI which provides non IB based connection management should be configured to run using hard coded SLs. It uses these SLs for every QP being opened.
5. ULPs that use CM interface (like SRP) have their own pre-assigned Service-ID and use it while obtaining PathRecord/MultiPathRecord (PR/MPR) for establishing connections. The SA receiving the PR/MPR matches it against the policy and returns the appropriate PR/MPR including SL, MTU, RATE and Lifetime.
6. ULPs and programs (e.g. SDP) use CMA to establish RC connection provide the CMA the target IP and port number. ULPs might also provide QoS-Class. The CMA then creates Service-ID for the ULP and passes this ID and optional QoS-Class in the PR/MPR request. The resulting PR/MPR is used for configuring the connection QP.

PathRecord and MultiPathRecord Enhancement for QoS:

As mentioned above, the PathRecord and MultiPathRecord attributes are enhanced to carry the Service-ID which is a 64bit value. A new field QoS-Class is also provided.

A new capability bit describes the SM QoS support in the SA class port info. This approach provides an easy migration path for existing access layer and ULPs by not introducing new set of PR/MPR attributes.

4.5.3 Supported Policy

The QoS policy, which is specified in a stand-alone file, is divided into the following four subsections:

I. Port Group

A set of CAs, Routers or Switches that share the same settings. A port group might be a partition defined by the partition manager policy, list of GUIDs, or list of port names based on NodeDescription.

II. Fabric Setup

Defines how the SL2VL and VLArb tables should be setup.

In OFED this part of the policy is ignored. SL2VL and VLArb tables should be configured in the OpenSM options file (opensm.opts).

III. QoS-Levels Definition

This section defines the possible sets of parameters for QoS that a client might be mapped to. Each set holds SL and optionally: Max MTU, Max Rate, Packet Lifetime and Path Bits.

Path Bits are not implemented in OFED.

IV. Matching Rules

A list of rules that match an incoming PR/MPR request to a QoS-Level. The rules are processed in order such as the first match is applied. Each rule is built out of a set of match expressions which should all match for the rule to apply. The matching expressions are defined for the following fields:

- SRC and DST to lists of port groups
- Service-ID to a list of Service-ID values or ranges
- QoS-Class to a list of QoS-Class values or ranges

4.5.4 CMA Features

The CMA interface supports Service-ID through the notion of port space as a prefix to the port number, which is part of the sockaddr provided to rdma_resolve_add(). The CMA also allows the ULP (like SDP) to propagate a request for a specific QoS-Class. The CMA uses the provided QoS-Class and Service-ID in the sent PR/MPR.

4.5.4.1 IPoIB

IPoIB queries the SA for its broadcast group information and uses the SL, MTU, RATE and Packet Lifetime available on the multicast group which forms this broadcast group.

4.5.4.2 SRP

The current SRP implementation uses its own CM callbacks (not CMA). So SRP fills in the Service-ID in the PR/MPR by itself and use that information in setting up the QP.

SRP Service-ID is defined by the SRP target I/O Controller (it also complies with IBTA Service-ID rules). The Service-ID is reported by the I/O Controller in the ServiceEntries DMA attribute and should be used in the PR/MPR if the SA reports its ability to handle QoS PR/MPRs.

4.5.5 OpenSM Features

The QoS related functionality that is provided by OpenSM—the Subnet Manager described in Chapter 8 can be split into two main parts:

I. Fabric Setup

During fabric initialization, the Subnet Manager parses the policy and apply its settings to the discovered fabric elements.

II. PR/MPR Query Handling

OpenSM enforces the provided policy on client request. The overall flow for such requests is: first the request is matched against the defined match rules such that the target QoS-Level definition is found. Given the QoS-Level a path(s) search is performed with the given restrictions imposed by that level.

4.6 Quality of Service Ethernet

4.6.1 Quality of Service Overview

Quality of Service (QoS) is a mechanism of assigning a priority to a network flow (socket, rdma_cm connection) and manage its guarantees, limitations and its priority over other flows. This is accomplished by mapping the user's priority to a hardware TC (traffic class) through a 2/3 stages process. The TC is assigned with the QoS attributes and the different flows behave accordingly

4.6.2 Mapping Traffic to Traffic Classes

Mapping traffic to TCs consists of several actions which are user controllable, some controlled by the application itself and others by the system/network administrators.

The following is the general mapping traffic to Traffic Classes flow:

1. The application sets the required Type of Service (ToS).
2. The ToS is translated into a Socket Priority (`sk_prio`).
3. The `sk_prio` is mapped to a User Priority (UP) by the system administrator (some applications set `sk_prio` directly).
4. The UP is mapped to TC by the network/system administrator.
5. TCs hold the actual QoS parameters

QoS can be applied on the following types of traffic. However, the general QoS flow may vary among them:

- **Plain Ethernet** - Applications use regular inet sockets and the traffic passes via the kernel Ethernet driver
- **RoCE** - Applications use the RDMA API to transmit using QPs
- **Raw Ethernet QP** - Application use VERBs API to transmit using a Raw Ethernet QP

4.6.3 Plain Ethernet Quality of Service Mapping

Applications use regular inet sockets and the traffic passes via the kernel Ethernet driver.

The following is the Plain Ethernet QoS mapping flow:

1. The application sets the ToS of the socket using `setsockopt` (`IP_TOS`, value).
2. ToS is translated into the `sk_prio` using a fixed translation:

```
TOS 0 <=> sk_prio 0
TOS 8 <=> sk_prio 2
TOS 24 <=> sk_prio 4
TOS 16 <=> sk_prio 6
```

3. The Socket Priority is mapped to the UP:
   - If the underlying device is a VLAN device, `egress_map` is used controlled by the `vconfig` command. This is per VLAN mapping.
   - If the underlying device is not a VLAN device, the `tc` command is used. In this case, even though `tc` manual states that the mapping is from the `sk_prio` to the TC number, the `mlx4_en` driver interprets this as a `sk_prio` to UP mapping.
     Mapping the sk_prio to the UP is done by using `tc_wrap.py -i <dev name> -u 0,1,2,3,4,5,6,7`
4. The UP is mapped to the TC as configured by the `mlnx_qos` tool or by the `lldpad` daemon if DCBX is used.

Socket applications can use `setsockopt` (`SK_PRIO`, value) to directly set the `sk_prio` of the socket. In this case the ToS to `sk_prio` fixed mapping is not needed. This allows the application and the administrator to utilize more than the 4 values possible via ToS.

In case of VLAN interface, the UP obtained according to the above mapping is also used in the VLAN tag of the traffic

4.6.4 RoCE Quality of Service Mapping

Applications use RDMA-CM API to create and use QPs.

The following is the RoCE QoS mapping flow:

1. The application sets the ToS of the QP using the `rdma_set_option` option (`RDMA_OPTION_ID_TOS`, value).
2. ToS is translated into the Socket Priority (`sk_prio`) using a fixed translation:

```
TOS 0 <=> sk_prio 0
TOS 8 <=> sk_prio 2
TOS 24 <=> sk_prio 4
TOS 16 <=> sk_prio 6
```

3. The Socket Priority is mapped to the User Priority (UP) using the `tc` command.
   - In case of a VLAN device where the parent real device is used for the purpose of this mapping
   - If the underlying device is a VLAN device, and the parent real device was not used for the mapping, the VLAN device's egress_map is used
   - On Kernels 3.13 and higher or Kernels that ported the functionality of enabled access to the VLAN device egress map (through vlan_dev_get_egress_qos_mask() call in if_vlan.h)

4. The UP is mapped to the TC as configured by the `mlnx_qos` tool or by the `lldpad` daemon if DCBX is used.

With RoCE, there can only be 4 predefined ToS values for the purpose of QoS mapping.

4.6.5 Raw Ethernet QP Quality of Service Mapping

Applications open a Raw Ethernet QP using VERBs directly.

The following is the RoCE QoS mapping flow:

1. The application sets the UP of the Raw Ethernet QP during the INIT to RTR state transition of the QP:
   - Sets `qp_attrs.ah_attrs.sl = up`
   - Calls `modify_qp` with `IB_QP_AV` set in the mask
2. The UP is mapped to the TC as configured by the `mlnx_qos` tool or by the `lldpad` daemon if DCBX is used

When using Raw Ethernet QP mapping, the TOS/sk_prio to UP mapping is lost.

Performing the Raw Ethernet QP mapping forces the QP to transmit using the given UP. If packets with VLAN tag are transmitted, UP in the VLAN tag will be overwritten with the given UP.

4.6.6 Map Priorities with tc_wrap.py/mlnx_qos

Network flow that can be managed by QoS attributes is described by a User Priority (UP). A user's `sk_prio` is mapped to UP which in turn is mapped into TC.

- Indicating the UP
  - When the user uses `sk_prio`, it is mapped into a UP by the `'tc'` tool. This is done by the `tc_wrap.py` tool which gets a list of <= 16 comma separated UP and maps the `sk_prio` to the specified UP.
    For example, `tc_wrap.py -ieth0 -u 1,5` maps `sk_prio 0` of `eth0` device to UP 1 and `sk_prio 1` to UP 5.
  - Setting `set_egress_map` in VLAN, maps the `skb_priority` of the VLAN to a `vlan_qos`. The `vlan_qos` is represents a UP for the VLAN device.
  - In RoCE, `rdma_set_option` with `RDMA_OPTION_ID_TOS` could be used to set the UP
  - When creating QPs, the `sl` field in `ibv_modify_qp` command represents the UP
- Indicating the TC
  - After mapping the `skb_priority` to UP, one should map the UP into a TC. This assigns the user priority to a specific hardware traffic class. In order to do that, `mlnx_qos` should

be used. `mlnx_qos` gets a list of a mapping between UPs to TCs. For example, `mlnx_qos -ieth0 -p 0,0,0,0,1,1,1,1` maps UPs 0-3 to `TC0`, and Ups 4-7 to `TC1`.

4.6.7 Quality of Service Properties

The different QoS properties that can be assigned to a TC are:

- Strict Priority (see "Strict Priority")
- Minimal Bandwidth Guarantee (ETS) (see "Minimal Bandwidth Guarantee (ETS)")
- Rate Limit (see "Rate Limit")

4.6.7.1 Strict Priority

When setting a TC's transmission algorithm to be 'strict', then this TC has absolute (strict) priority over other TC strict priorities coming before it (as determined by the TC number: TC 7 is highest priority, TC 0 is lowest). It also has an absolute priority over non strict TCs (ETS).

This property needs to be used with care, as it may easily cause starvation of other TCs.

A higher strict priority TC is always given the first chance to transmit. Only if the highest strict priority TC has nothing more to transmit, will the next highest TC be considered.

Non strict priority TCs will be considered last to transmit.

This property is extremely useful for low latency low bandwidth traffic. Traffic that needs to get immediate service when it exists, but is not of high volume to starve other transmitters in the system.

4.6.7.2 Minimal Bandwidth Guarantee (ETS)

After servicing the strict priority TCs, the amount of bandwidth (BW) left on the wire may be split among other TCs according to a minimal guarantee policy.

If, for instance, TC0 is set to 80% guarantee and TC1 to 20% (the TCs sum must be 100), then the BW left after servicing all strict priority TCs will be split according to this ratio.

Since this is a minimal guarantee, there is no maximum enforcement. This means, in the same example, that if TC1 did not use its share of 20%, the reminder will be used by TC0.

4.6.7.3 Rate Limit

Rate limit defines a maximum bandwidth allowed for a TC. Please note that 10% deviation from the requested values is considered acceptable.

4.6.8 Quality of Service Tools

4.6.8.1 mlnx_qos

`mlnx_qos` is a centralized tool used to configure QoS features of the local host. It communicates directly with the driver thus does not require setting up a DCBX daemon on the system.

The `mlnx_qos` tool enables the administrator of the system to:

- Inspect the current QoS mappings and configuration

  The tool will also display maps configured by TC and `vconfig set_egress_map` tools, in order to give a centralized view of all QoS mappings.
- Set UP to TC mapping

- Assign a transmission algorithm to each TC (strict or ETS)
- Set minimal BW guarantee to ETS TCs
- Set rate limit to TCs

For unlimited ratelimit set the ratelimit to 0.

Usage:

```
mlnx_qos -i <interface> [options]
```

Options:

```
  --version             show program's version number and exit
  -h, --help            show this help message and exit
  -p LIST, --prio_tc=LIST
                        maps UPs to TCs. LIST is 8 comma seperated TC numbers.
                        Example: 0,0,0,0,1,1,1,1 maps UPs 0-3 to TC0, and UPs
                        4-7 to TC1
  -s LIST, --tsa=LIST   Transmission algorithm for each TC. LIST is comma
                        seperated algorithm names for each TC. Possible
                        algorithms: strict, etc. Example: ets,strict,ets sets
                        TC0,TC2 to ETS and TC1 to strict. The rest are
                        unchanged.
  -t LIST, --tcbw=LIST  Set minimal guaranteed %BW for ETS TCs. LIST is comma
                        seperated percents for each TC. Values set to TCs that
                        are not configured to ETS algorithm are ignored, but
                        must be present. Example: if TC0,TC2 are set to ETS,
                        then 10,0,90 will set TC0 to 10% and TC2 to 90%.
                        Percents must sum to 100.
  -r LIST, --ratelimit=LIST
                        Rate limit for TCs (in Gbps). LIST is a comma
                        seperated Gbps limit for each TC. Example: 1,8,8 will
                        limit TC0 to 1Gbps, and TC1,TC2 to 8 Gbps each.
  -i INTF, --interface=INTF
                        Interface name
  -a                    Show all interface's TCs
```

Get Current Configuration:

```
tc: 0 ratelimit: unlimited, tsa: strict
         up:  0
                 skprio: 0
                 skprio: 1
                 skprio: 2 (tos: 8)
                 skprio: 3
                 skprio: 4 (tos: 24)
                 skprio: 5
                 skprio: 6 (tos: 16)
                 skprio: 7
                 skprio: 8
                 skprio: 9
                 skprio: 10
                 skprio: 11
                 skprio: 12
                 skprio: 13
                 skprio: 14
                 skprio: 15
         up:  1
         up:  2
         up:  3
         up:  4
         up:  5
         up:  6
         up:  7
```

Set ratelimit. 3Gbps for tc0 4Gbps for tc1 and 2Gbps for tc2:

```
tc: 0 ratelimit: 3 Gbps, tsa: strict
         up:  0
                 skprio: 0
                 skprio: 1
                 skprio: 2 (tos: 8)
                 skprio: 3
                 skprio: 4 (tos: 24)
                 skprio: 5
                 skprio: 6 (tos: 16)
                 skprio: 7
                 skprio: 8
                 skprio: 9
                 skprio: 10
                 skprio: 11
                 skprio: 12
                 skprio: 13
                 skprio: 14
                 skprio: 15
         up:  1
         up:  2
         up:  3
         up:  4
         up:  5
         up:  6
         up:  7
```

Configure QoS. map UP 0,7 to tc0, 1,2,3 to tc1 and 4,5,6 to tc 2. set tc0,tc1 as ets and tc2 as strict. divide ets 30% for tc0 and 70% for tc1:

```
mlnx_qos -i eth3 -s ets,ets,strict -p 0,1,1,1,2,2,2 -t 30,70
tc: 0 ratelimit: 3 Gbps, tsa: ets, bw: 30%
         up:  0
                 skprio: 0
                 skprio: 1
                 skprio: 2 (tos: 8)
                 skprio: 3
                 skprio: 4 (tos: 24)
                 skprio: 5
                 skprio: 6 (tos: 16)
                 skprio: 7
                 skprio: 8
                 skprio: 9
                 skprio: 10
                 skprio: 11
                 skprio: 12
                 skprio: 13
                 skprio: 14
                 skprio: 15
  up:  7
tc: 1 ratelimit: 4 Gbps, tsa: ets, bw: 70%
```

```
         up:  1
         up:  2
         up:  3
tc: 2 ratelimit: 2 Gbps, tsa: strict
         up:  4
         up:  5
         up:  6
```

4.6.8.2 tc and tc_wrap.py

The 'tc' tool is used to setup `sk_prio` to UP mapping, using the `mqprio` queue discipline.

In kernels that do not support `mqprio` (such as 2.6.34), an alternate mapping is created in `sysfs`. The 'tc_wrap.py' tool will use either the `sysfs` or the 'tc' tool to configure the `sk_prio` to UP mapping.

Usage:

```
tc_wrap.py -i <interface> [options]
```

Options:

```
--version            show program's version number and exit
-h, --help           show this help message and exit
-u SKPRIO_UP, --skprio_up=SKPRIO_UP  maps sk_prio to UP. LIST is <=16 comma separated
                                     UP. index of element is sk_prio.
-i INTF, --interface=INTF            Interface name
```

Example: set skprio 0-2 to UP0, and skprio 3-7 to UP1 on eth4

```
UP  0
        skprio: 0
        skprio: 1
        skprio: 2 (tos: 8)
        skprio: 7
        skprio: 8
        skprio: 9
        skprio: 10
        skprio: 11
        skprio: 12
        skprio: 13
        skprio: 14
        skprio: 15
UP  1
        skprio: 3
        skprio: 4 (tos: 24)
        skprio: 5
        skprio: 6 (tos: 16)
```

```
UP  2
UP  3
UP  4
UP  5
UP  6
UP  7
```

4.6.8.3 Additional Tools

tc tool compiled with the `sch_mqprio` module is required to support kernel v2.6.32 or higher. This is a part of `iproute2` package v2.6.32-19 or higher. Otherwise, an alternative custom sysfs interface is available.

- `mlnx_qos tool` (package: ofed-scripts) requires python >= 2.5
- `tc_wrap.py` (package: ofed-scripts) requires python >= 2.5

4.7 Ethernet Time-Stamping

4.7.1 Ethernet Time-Stamping Service

Time Stamping is currently supported in ConnectX®-3/ConnectX®-3 Pro adapter cards only.

Time stamping is the process of keeping track of the creation of a packet. A time-stamping service supports assertions of proof that a datum existed before a particular time. Incoming packets are time-stamped before they are distributed on the PCI depending on the congestion in the PCI buffers. Outgoing packets are time-stamped very close to placing them on the wire.

4.7.1.1 Enabling Time Stamping

Time-stamping is off by default and should be enabled before use.

➢ ***To enable time stamping for a socket:***

- Call setsockopt() with SO_TIMESTAMPING and with the following flags:

```
SOF_TIMESTAMPING_TX_HARDWARE:  try to obtain send time stamp in hardware
SOF_TIMESTAMPING_TX_SOFTWARE:  if SOF_TIMESTAMPING_TX_HARDWARE is off or
                               fails, then do it in software
SOF_TIMESTAMPING_RX_HARDWARE:  return the original, unmodified time stamp
                               as generated by the hardware
SOF_TIMESTAMPING_RX_SOFTWARE:  if SOF_TIMESTAMPING_RX_HARDWARE is off or
                               fails, then do it in software
SOF_TIMESTAMPING_RAW_HARDWARE: return original raw hardware time stamp
SOF_TIMESTAMPING_SYS_HARDWARE: return hardware time stamp transformed to
                               the system time base
SOF_TIMESTAMPING_SOFTWARE:     return system time stamp generated in
                               software
SOF_TIMESTAMPING_TX/RX determine how time stamps are generated.
SOF_TIMESTAMPING_RAW/SYS determine how they are reported
```

- ***To enable time stamping for a net device:***

  Admin privileged user can enable/disable time stamping through calling ioctl (sock, SIOCSH-WTSTAMP, &ifreq) with following values:

  Send side time sampling:

  - Enabled by ifreq.hwtstamp_config.tx_type when

```
/* possible values for hwtstamp_config->tx_type */
enum hwtstamp_tx_types {
        /*
         * No outgoing packet will need hardware time stamping;
         * should a packet arrive which asks for it, no hardware
         * time stamping will be done.
        */
        HWTSTAMP_TX_OFF,

        /*
         * Enables hardware time stamping for outgoing packets;
         * the sender of the packet decides which are to be
         * time stamped by setting %SOF_TIMESTAMPING_TX_SOFTWARE
         * before sending the packet.
         */
        HWTSTAMP_TX_ON,
  /*
         * Enables time stamping for outgoing packets just as
         * HWTSTAMP_TX_ON does, but also enables time stamp insertion
         * directly into Sync packets. In this case, transmitted Sync
         * packets will not received a time stamp via the socket error
         * queue.
         */
        HWTSTAMP_TX_ONESTEP_SYNC,
};
Note: for send side time stamping currently only HWTSTAMP_TX_OFF and
HWTSTAMP_TX_ON are supported.
```

Receive side time sampling:

- Enabled by ifreq.hwtstamp_config.rx_filter when

```
/* possible values for hwtstamp_config->rx_filter */
enum hwtstamp_rx_filters {
        /* time stamp no incoming packet at all */
        HWTSTAMP_FILTER_NONE,

        /* time stamp any incoming packet */
        HWTSTAMP_FILTER_ALL,
      /* return value: time stamp all packets requested plus some others */
        HWTSTAMP_FILTER_SOME,

        /* PTP v1, UDP, any kind of event packet */
        HWTSTAMP_FILTER_PTP_V1_L4_EVENT,
        /* PTP v1, UDP, Sync packet */
        HWTSTAMP_FILTER_PTP_V1_L4_SYNC,
        /* PTP v1, UDP, Delay_req packet */
        HWTSTAMP_FILTER_PTP_V1_L4_DELAY_REQ,
        /* PTP v2, UDP, any kind of event packet */
        HWTSTAMP_FILTER_PTP_V2_L4_EVENT,
        /* PTP v2, UDP, Sync packet */
        HWTSTAMP_FILTER_PTP_V2_L4_SYNC,
        /* PTP v2, UDP, Delay_req packet */
        HWTSTAMP_FILTER_PTP_V2_L4_DELAY_REQ,

        /* 802.AS1, Ethernet, any kind of event packet */
        HWTSTAMP_FILTER_PTP_V2_L2_EVENT,
        /* 802.AS1, Ethernet, Sync packet */
        HWTSTAMP_FILTER_PTP_V2_L2_SYNC,
        /* 802.AS1, Ethernet, Delay_req packet */
        HWTSTAMP_FILTER_PTP_V2_L2_DELAY_REQ,

        /* PTP v2/802.AS1, any layer, any kind of event packet */
        HWTSTAMP_FILTER_PTP_V2_EVENT,
        /* PTP v2/802.AS1, any layer, Sync packet */
        HWTSTAMP_FILTER_PTP_V2_SYNC,
        /* PTP v2/802.AS1, any layer, Delay_req packet */
        HWTSTAMP_FILTER_PTP_V2_DELAY_REQ,
};
Note: for receive side time stamping currently only HWTSTAMP_FILTER_NONE and
HWTSTAMP_FILTER_ALL are supported.
```

4.7.1.2 Getting Time Stamping

Once time stamping is enabled time stamp is placed in the socket Ancillary data. recvmsg() can be used to get this control message for regular incoming packets. For send time stamps the outgoing packet is looped back to the socket's error queue with the send time stamp(s) attached. It can be received with recvmsg(flags=MSG_ERRQUEUE). The call returns the original outgoing packet data including all headers preprended down to and including the link layer, the scm_timestamping control message and a sock_extended_err control message with ee_errno==ENOMSG and ee_origin==SO_EE_ORIGIN_TIMESTAMPING. A socket with such

a pending bounced packet is ready for reading as far as select() is concerned. If the outgoing packet has to be fragmented, then only the first fragment is time stamped and returned to the sending socket.

When time-stamping is enabled, VLAN stripping is disabled. For more info please refer to Documentation/networking/timestamping.txt in kernel.org

4.7.1.3 Querying Time Stamping Capabilities via ethtool

➢ ***To display Time Stamping capabilities via ethtool:***

- Show Time Stamping capabilities

```
ethtool -T eth<x>
```

Example:

```
ethtool -T eth0
Time stamping parameters for p2p1:
Capabilities:
        hardware-transmit     (SOF_TIMESTAMPING_TX_HARDWARE)
        software-transmit     (SOF_TIMESTAMPING_TX_SOFTWARE)
        hardware-receive      (SOF_TIMESTAMPING_RX_HARDWARE)
        software-receive      (SOF_TIMESTAMPING_RX_SOFTWARE)
        software-system-clock (SOF_TIMESTAMPING_SOFTWARE)
        hardware-raw-clock    (SOF_TIMESTAMPING_RAW_HARDWARE)
PTP Hardware Clock: 1
Hardware Transmit Timestamp Modes:
        off                   (HWTSTAMP_TX_OFF)
        on                    (HWTSTAMP_TX_ON)
Hardware Receive Filter Modes:
        none                  (HWTSTAMP_FILTER_NONE)
        all                   (HWTSTAMP_FILTER_ALL)
```

For more details on PTP Hardware Clock, please refer to:

https://www.kernel.org/doc/Documentation/ptp/ptp.txt

4.7.2 RoCE Time Stamping

RoCE Time Stamping is currently at beta level.
Please be aware that everything listed here is subject to change.

RoCE Time Stamping allows you to stamp packets when they are sent to the wire / received from the wire. The time stamp is given in a raw hardware cycles, but could be easily converted into hardware referenced nanoseconds based time. Additionally, it enables you to query the hardware for the hardware time, thus stamp other application's event and compare time.

4.7.2.1 Query Capabilities

Time stamping is available if and only the hardware reports it is capable of reporting it. To verify whether RoCE Time Stamping is available, run `ibv_exp_query_device`.

For example:

```
struct ibv_exp_device_attr attr;
ibv_exp_query_device(context, &attr);
if (attr.comp_mask & IBV_EXP_DEVICE_ATTR_WITH_TIMESTAMP_MASK) {
if (attr.timestamp_mask) {
                /* Time stamping is supported with mask attr.timestamp_mask */
        }
}
if (attr.comp_mask & IBV_EXP_DEVICE_ATTR_WITH_HCA_CORE_CLOCK) {
        if (attr.hca_core_clock) {
                /* reporting the device's clock is supported. */
                /* attr.hca_core_clock is the frequency in MHZ */
        }
}
```

4.7.2.2 Creating Time Stamping Completion Queue

To get time stamps, a suitable extended Completion Queue (CQ) must be created via a special call to `ibv_exp_create_cq` verb.

```
cq_init_attr.flags = IBV_EXP_CQ_TIMESTAMP;
cq_init_attr.comp_mask = IBV_EXP_CQ_INIT_ATTR_FLAGS;
cq = ibv_exp_create_cq(context, cqe, node, NULL, 0, &cq_init_attr);
```

This CQ cannot report SL or SLID information. The value of `sl` and `sl_id` fields in `struct ibv_exp_wc` are invalid. Only the fields indicated by the `exp_wc_flags` field in `struct ibv_exp_wc` contains a valid and usable value.

When using Time Stamping, several fields of `struct ibv_exp_wc` are not available resulting in RoCE UD / RoCE traffic with VLANs failure.

4.7.2.3 Polling a Completion Queue

Polling a CQ for time stamp is done via the `ibv_exp_poll_cq` verb.

```
ret = ibv_exp_poll_cq(cq, 1, &wc_ex, sizeof(wc_ex));
if (ret > 0) {
                /* CQ returned a wc */
                if (wc_ex.exp_wc_flags & IBV_EXP_WC_WITH_TIMESTAMP) {
                /* This wc contains a timestamp */
                timestamp = wc_ex.timestamp;
                /* Timestamp is given in raw hardware time */
                }
}
```

CQs that are opened with the `ibv_exp_create_cq` verbs should be always be polled with the `ibv_exp_poll_cq` verb.

4.7.2.4 Querying the Hardware Time

Querying the hardware for time is done via the `ibv_exp_query_values` verb.

For example:

```
ret = ibv_exp_query_values(context, IBV_EXP_VALUES_HW_CLOCK, &queried_values);
if (!ret && queried_values.comp_mask & IBV_EXP_VALUES_HW_CLOCK)
queried_time = queried_values.hwclock;
```

To change the queried time in nanoseconds resolution, use the `IBV_EXP_VALUES_HW_CLOCK_NS` flag along with the `hwclock_ns` field.

```
ret = ibv_exp_query_values(context, IBV_EXP_VALUES_HW_CLOCK_NS, &queried_values);
if (!ret && queried_values.comp_mask & IBV_EXP_VALUES_HW_CLOCK_NS)
queried_time_ns = queried_values.hwclock_ns;
```

Querying the Hardware Time is available only on physical functions / native machines.

4.8 Ethernet VXLAN

4.8.1 Prerequisites

- HCA: ConnectX-3 Pro
- Firmware 2.31.5020 or higher
- RHEL7, Ubuntu 14.04 or upstream kernel 3.12.10 (or higher)
- DMFS enabled

4.8.2 Enabling VXLAN

To enable the vxlan offloads support load the `mlx4_core` driver with Device-Managed Flow-steering (DMFS) enabled. Best would be to create an `/etc/modprobe.d/mlx4.conf` file with the following contents:

```
# enable DMFS --> enable VXLAN offloads on CX3-pro
options mlx4_core log_num_mgm_entry_size=-1
```

To verify VXLAN is enabled, verify the Ethernet net-device created by the mlx4_en driver advertises the `NETIF_F_GSO_UDP_TUNNEL` feature, which can be seen by the `"ethtool -K $DEV | grep udp"`

For example:

```
$ ethtool -k eth0 | grep udp_tnl
tx-udp_tnl-segmentation: on
```

Make sure that the VXLAN tunnel is set over UDP port 4789, which is the ConnectX firmware default. If using standard Linux bridge (and not open vswitch) set the following `/etc/modprobe.d/vxlan.conf`:

```
options vxlan udp_port=4789
```

4.8.3 Important Notes

- VXLAN tunneling adds 50 bytes (14-eth + 20-ip + 8-udp + 8-vxlan) to the VM Ethernet frame. Please verify that either the MTU of the NIC who sends the packets, e.g. the VM virtio-net NIC or the host side veth device or the uplink takes into account the tunneling overhead. Meaning, the MTU of the sending NIC has to be decremented by 50 bytes (e.g 1450 instead of 1500), or the uplink NIC MTU has to be incremented by 50 bytes (e.g 1550 instead of 1500)
- From upstream 3.15-rc1 and onward, it is possible to use arbitrary UDP port for VXLAN. Note that this requires firmware version 2.31.2800 or higher. Additionally, you need to enable this kernel configuration option `CONFIG_MLX4_EN_VXLAN=y`.
- On upstream kernels 3.12/3.13 GRO with VXLAN is not supported

4.9 Atomic Operations

4.9.1 Atomic Operations in mlx5 Driver

Atomic Operations in Connect-IB® (mlx5 driver) are fully supported on big-endian machines (e.g. PPC). Their support is limited on little-endian machines (e.g. x86)

When using `ibv_exp_query_device` on little-endian machines with Connect-IB® the `attr.exp_atomic_cap` is set to `IBV_EXP_ATOMIC_HCA_REPLY_BE` which indicates that if enabled, the atomic operation replied value is big-endian and contradicts the host endianness.

To enable atomic operation with this endianness contradiction use the `ibv_exp_create_qp` to create the QP and set the `IBV_EXP_QP_CREATE_ATOMIC_BE_REPLY` flag on `exp_create_flags`.

4.9.2 Enhanced Atomic Operations

ConnectX® implements a set of Extended Atomic Operations beyond those defined by the IB spec. Atomicity guarantees, Atomic Ack generation, ordering rules and error behavior for this set of extended Atomic operations is the same as that for IB standard Atomic operations (as defined in section 9.4.5 of the IB spec).

4.9.2.1 Masked Compare and Swap (MskCmpSwap)

The MskCmpSwap atomic operation is an extension to the CmpSwap operation defined in the IB spec. MskCmpSwap allows the user to select a portion of the 64 bit target data for the "compare" check as well as to restrict the swap to a (possibly different) portion. The pseudocode below describes the operation:

```
| atomic_response = *va
| if (!((compare_add ^ *va) & compare_add_mask)) then
|     *va = (*va & ~(swap_mask)) | (swap & swap_mask)
|
| return atomic_response
```

The additional operands are carried in the Extended Transport Header. Atomic response generation and packet format for MskCmpSwap is as for standard IB Atomic operations.

4.9.2.2 Masked Fetch and Add (MFetchAdd)

The MFetchAdd Atomic operation extends the functionality of the standard IB FetchAdd by allowing the user to split the target into multiple fields of selectable length. The atomic add is done independently on each one of this fields. A bit set in the field_boundary parameter specifies the field boundaries. The pseudocode below describes the operation:

```
| bit_adder(ci, b1, b2, *co)
| {
|   value = ci + b1 + b2
|   *co = !!(value & 2)
|
|   return value & 1
| }
|
| #define MASK_IS_SET(mask, attr)      (!!((mask)&(attr)))
| bit_position = 1
| carry = 0
| atomic_response = 0
|
| for i = 0 to 63
| {
|         if ( i != 0 )
|                 bit_position =  bit_position << 1
|
|         bit_add_res = bit_adder(carry, MASK_IS_SET(*va, bit_position),
|                                 MASK_IS_SET(compare_add, bit_position), &new_carry)
|         if (bit_add_res)
|                 atomic_response |= bit_position
|
|         carry = ((new_carry) && (!MASK_IS_SET(compare_add_mask, bit_position)))
| }
|
| return atomic_response
```

4.10 Ethernet Tunneling Over IPoIB Driver (eIPoIB)

The eth_ipoib driver provides a standard Ethernet interface to be used as a Physical Interface (PIF) into the Hypervisor virtual network, and serves one or more Virtual Interfaces (VIF). This driver supports L2 Switching (Direct Bridging) as well as other L3 Switching modes (e.g. NAT). This document explains the configuration and driver behavior when configured in Bridging mode.

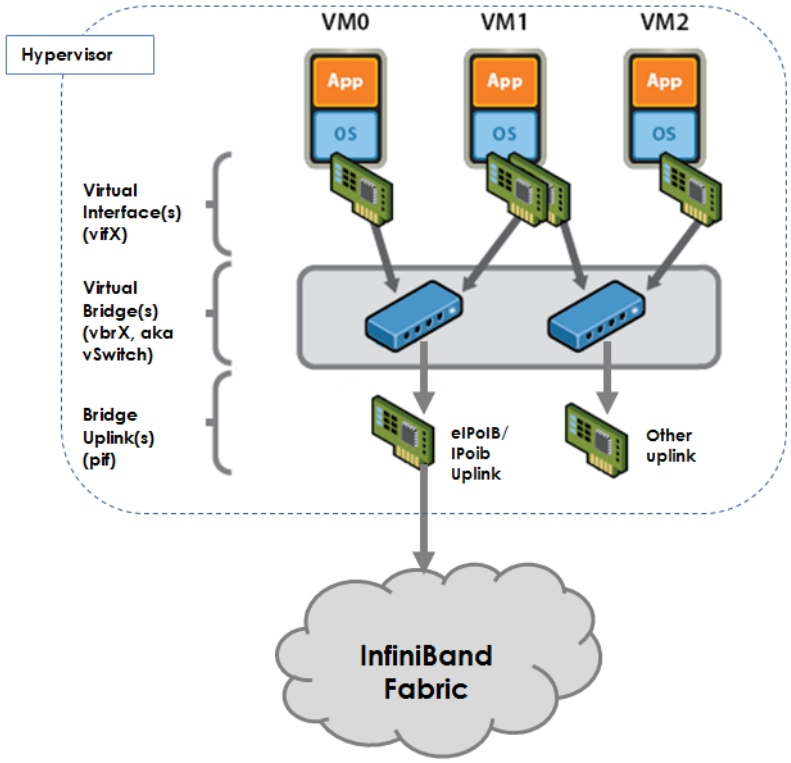
In virtualization environment, a virtual machine can be expose to the physical network by performing the next setting:

Step 1. Create a virtual bridge

Step 2. Attach the para-virtualized interface created by the eth_ipoib driver to the bridge

Step 3. Attach the Ethernet interface in the Virtual Machine to that bridge

The diagram below describes the topology that was created after these steps:



The diagram shows how the traffic from the Virtual Machine goes to the virtual-bridge in the Hypervisor and from the bridge to the eIPoIB interface. eIPoIB interface is the Ethernet interface that enslaves the IPoIB interfaces in order to send/receive packets from the Ethernet interface in the Virtual Machine to the IB fabric beneath.

4.10.1 Enabling the eIPoIB Driver

Once the mlnx_ofed driver installation is completed, perform the following:

Step 1. Open the `/etc/infiniband/openib.conf` file and include:

```
E_IPOIB_LOAD=yes
```

Step 2. Restart the InfiniBand drivers.

```
/etc/init.d/openibd restart
```

4.10.2 Configuring the Ethernet Tunneling Over IPoIB Driver

When eth_ipoib is loaded, number of eIPoIB interfaces are created, with the following default naming scheme: `ethX`, where X represents the ETH port available on the system.

To check which eIPoIB interfaces were created:

```
cat /sys/class/net/eth_ipoib_interfaces
```

For example, on a system with dual port HCA, the following two interfaces might be created; `eth4` and `eth5`.

```
cat /sys/class/net/eth_ipoib_interfaces
eth4 over IB port: ib0
eth5 over IB port: ib1
```

These interfaces can be used to configure the network for the guest. For example, if the guest has a VIF that is connected to the Virtual Bridge `br0`, then enslave the eIPoIB interface to `br0` by running:

```
# brctl addif br0 ethX
```

In RHEL KVM environment, there are other methods to create/configure your virtual network (e.g. macvtap). For additional information, please refer to the Red Hat User Manual.

The IPoIB daemon (ipoibd) detects the new virtual interface that is attached to the same bridge as the eIPoIB interface and creates a new IPoIB instances for it in order to send/receive data. As a result, number of IPoIB interfaces (ibX.Y) are shown as being created/destroyed, and are being enslaved to the corresponding `ethX` interface to serve any active VIF in the system according to the set configuration, This process is done automatically by the `ipoibd` service.

- ***To see the list of IPoIB interfaces enslaved under eth_ipoib interface.***

```
# cat /sys/class/net/ethX/eth/vifs
```

For example:

```
# cat /sys/class/net/eth5/eth/vifs
  SLAVE=ib0.1      MAC=9a:c2:1f:d7:3b:63 VLAN=N/A
  SLAVE=ib0.2      MAC=52:54:00:60:55:88 VLAN=N/A
  SLAVE=ib0.3      MAC=52:54:00:60:55:89 VLAN=N/A
```

Each ethX interface has at lease one ibX.Y slave to serve the PIF itself. In the VIFs list of ethX you will notice that ibX.1 is always created to serve applications running from the Hypervisor on top of the ethX interface directly.

For InfiniBand applications that require native IPoIB interfaces (e.g. CMA), the original IPoIB interfaces ibX can still be used. For example, CMA and ethX drivers can co-exist and make use of IPoIB ports; CMA can use ib0, while eth0.ipoib interface will use ibX.Y interfaces.

- ***To see the list of eIPoIB interfaces.***

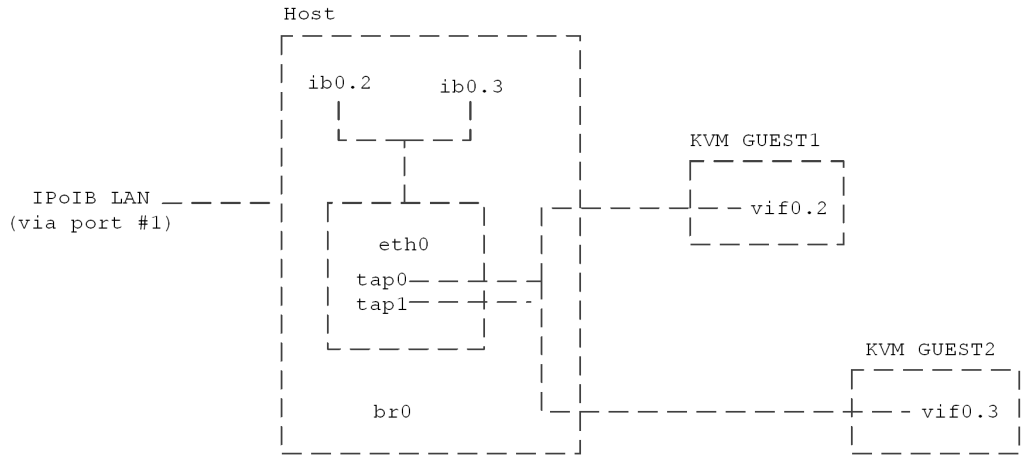
```
# cat /sys/class/net/eth_ipoib_interfaces
```

For example:

```
# cat /sys/class/net/eth_ipoib_interfaces
  eth4 over IB port: ib0
  eth5 over IB port: ib1
```

The example above shows, two eIPoIB interfaces, where eth4 runs traffic over ib0, and eth5 runs traffic over ib1.

Figure 3: An Example of a Virtual Network



The example above shows a few IPoIB instances that serve the virtual interfaces at the Virtual Machines.

To display the services provided to the Virtual Machine interfaces:

```
# cat /sys/class/net/eth0/eth/vifs
```

Example:

```
# cat /sys/class/net/eth0/eth/vifs
  SLAVE=ib0.2 MAC=52:54:00:60:55:88 VLAN=N/A
```

In the example above the ib0.2 IPoIB interface serves the MAC 52:54:00:60:55:88 with no VLAN tag for that interface.

4.10.3 VLAN Configuration Over an eIPoIB Interface

eIPoIB driver supports VLAN Switch Tagging (VST) mode, which enables the virtual machine interface to have no VLAN tag over it, thus allowing VLAN tagging to be handled by the Hypervisor.

➢ ***To attach a Virtual Machine interface to a specific isolated tag:***

Step 1. Verify the VLAN tag to be used has the same pkey value that is already configured on that ib port.

```
# cat /sys/class/infiniband/mlx4_0/ports/<ib port>/pkeys/*
```

Step 2. Create a VLAN interface in the Hypervisor, over the eIPoIB interface.

```
# vconfig add <eIPoIB interface> <vlan tag>
```

Step 3. Attach the new VLAN interface to the same bridge that the virtual machine interface is already attached to.

```
# brctl addif <br-name> <interface-name>
```

For example, to create the VLAN tag 3 with pkey 0x8003 over that port in the eIPoIB interface eth4, run:

```
#vconfig add eth4 3
#brctl addif br2 eth4.3
```

4.10.4 Setting Performance Tuning

- Use 4K MTU over OpenSM. For further information, please refer to Section 8.4.1, “File Format”, on page 173

```
Default=0xffff, ipoib, mtu=5 : ALL=full;
```

- Use MTU for 4K (4092 Bytes):
  - In UD mode, the maximum MTU value is 4092 Bytes

  Make sure that all interfaces (including the guest interface and its virtual bridge) have the same MTU value (MTU 4092 Bytes). For further information of MTU settings, please refer to the Hypervisor User Manual.
- Tune the TCP/IP stack using sysctl (dom0/domu)

```
# /sbin/sysctl_perf_tuning
```

- Other performance tuning for KVM environment such as vCPU pinning and NUMA tuning may apply. For further information, please refer to the Hypervisor User Manual.

4.11 Contiguous Pages

Contiguous Pages improves performance by allocating user memory regions over physical contiguous pages. It enables a user application to ask low level drivers to allocate contiguous memory for it as part of `ibv_reg_mr`.

Additional performance improvements can be reached by allocating Queue Pair (QP) and Completion Queue (CQ|) buffers to the Contiguous Pages.

To activate set the below environment variables with values of PREFER_CONTIG or CONTIG.

- For QP: MLX_QP_ALLOC_TYPE
- For CQ: MLX_CQ_ALLOC_TYPE

The following are all the possible values that can be allocated to the buffer:

Table 5 - Buffer Values

Possible Value[1]	Description
ANON	Use current pages ANON small ones.
HUGE	Force huge pages.
CONTIG	Force contiguous pages.
PREFER_CONTIG	Try contiguous fallback to ANON small pages. (Default)
PREFER_HUGE	Try huge fallback to ANON small pages.
ALL	Try huge fallback to contiguous if failed fallback to ANON small pages.

1. Values are NOT case sensitive.

Usage:

The application calls the `ibv_exp_reg_mr` API which turns on the `IBV_EXP_ACCESS_ALLOCATE_MR` bit and sets the input address to NULL. Upon success, the

address field of the struct `ibv_mr` will hold the address to the allocated memory block. This block will be freed implicitly when the `ibv_dereg_mr()` is called.

The following are environment variables that can be used to control error cases / contiguity:

Table 6 - Parameters Used to Control Error Cases / Contiguity

Parameters	Description
`MLX_MR_ALLOC_TYPE`	Configures the allocator type. • `ALL` (Default) - Uses all possible allocator and selects most efficient allocator. • `ANON` - Enables the usage of anonymous pages and disables the allocator • `CONTIG` - Forces the usage of the contiguous pages allocator. If contiguous pages are not available the allocation fails
`MLX_MR_MAX_LOG2_CONTIG_BSIZE`	Sets the maximum contiguous block size order. • Values: 12-23 • Default: 23
`MLX_MR_MIN_LOG2_CONTIG_BSIZE`	Sets the minimum contiguous block size order. • Values: 12-23 • Default: 12

4.12 Shared Memory Region

Shared Memory Region is only applicable to the mlx4 driver.

Shared Memory Region (MR) enables sharing MR among applications by implementing the "Register Shared MR" verb which is part of the IB spec.

Sharing MR involves the following steps:

Step 1. Request to create a shared MR

The application sends a request via the `ibv_exp_reg_mr` API to create a shared MR. The application supplies the allowed sharing access to that MR. If the MR was created successfully, a unique MR ID is returned as part of the struct `ibv_mr` which can be used by other applications to register with that MR.

The underlying physical pages must not be Least Recently Used (LRU) or Anonymous. To disable that, you need to turn on the `IBV_EXP_ACCESS_ALLOCATE_MR` bit as part of the sharing bits.

Usage:

- Turns on via the `ibv_exp_reg_mr` one or more of the sharing access bits. The sharing bits are part of the `ibv_exp_reg_mr` man page.
- Turns on the `IBV_EXP_ACCESS_ALLOCATE_MR` bit

Step 2. Request to register to a shared MR

A new verb called `ibv_exp_reg_shared_mr` is added to enable sharing an MR. To use this verb, the application supplies the MR ID that it wants to register for and the desired access

mode to that MR. The desired access is validated against its given permissions and upon successful creation, the physical pages of the original MR are shared by the new MR. Once the MR is shared, it can be used even if the original MR was destroyed.

The request to share the MR can be repeated multiple times and an arbitrary number of Memory Regions can potentially share the same physical memory locations.

Usage:

- Uses the “handle” field that was returned from the `ibv_exp_reg_mr` as the `mr_handle`
- Supplies the desired “access mode” for that MR
- Supplies the address field which can be either NULL or any hint as the required output. The address and its length are returned as part of the `ibv_mr struct`.

  To achieve high performance it is highly recommended to supply an address that is aligned as the original memory region address. Generally, it may be an alignment to 4M address.

For further information on how to use the `ibv_exp_reg_shared_mr` verb, please refer to the `ibv_exp_reg_shared_mr` man page and/or to the `ibv_shared_mr` sample program which demonstrates a basic usage of this verb.
Further information on the `ibv_shared_mr` sample program can be found in the `ibv_shared_mr` man page.

4.13 XRC - eXtended Reliable Connected Transport Service for InfiniBand

XRC allows significant savings in the number of QPs and the associated memory resources required to establish all to all process connectivity in large clusters.

It significantly improves the scalability of the solution for large clusters of multicore end-nodes by reducing the required resources.

For further details, please refer to the "Annex A14 Supplement to InfiniBand Architecture Specification Volume 1.2.1"

A new API can be used by user space applications to work with the XRC transport. The legacy API is currently supported in both binary and source modes, however it is deprecated. Thus we recommend using the new API.

The new verbs to be used are:

- `ibv_open_xrcd/ibv_close_xrcd`
- `ibv_create_srq_ex`
- `ibv_get_srq_num`
- `ibv_create_qp_ex`
- `ibv_open_qp`

Please use ibv_xsrq_pingpong for basic tests and code reference. For detailed information regarding the various options for these verbs, please refer to their appropriate man pages.

4.14 Flow Steering

Flow Steering is applicable to the mlx4 driver only.

Flow steering is a new model which steers network flows based on flow specifications to specific QPs. Those flows can be either unicast or multicast network flows. In order to maintain flexibil-

ity, domains and priorities are used. Flow steering uses a methodology of flow attribute, which is a combination of L2-L4 flow specifications, a destination QP and a priority. Flow steering rules may be inserted either by using ethtool or by using InfiniBand verbs. The verbs abstraction uses a different terminology from the flow attribute (`ibv_exp_flow_attr`), defined by a combination of specifications (`struct ibv_exp_flow_spec_*`).

4.14.1 Enable/Disable Flow Steering

Flow Steering is disabled by default and regular L2 steering is performed instead (B0 Steering). When using SR-IOV, flow steering is enabled if there is an adequate amount of space to store the flow steering table for the guest/master.

➢ ***To enable Flow Steering:***

Step 1. Open the `/etc/modprobe.d/mlnx.conf` file.

Step 2. Set the parameter `log_num_mgm_entry_size` to `-1` by writing the option `mlx4_core log_num_mgm_entry_size=-1`.

Step 3. Restart the driver

➢ ***To disable Flow Steering:***

Step 1. Open the `/etc/modprobe.d/mlnx.conf` file.

Step 2. Remove the `options mlx4_core log_num_mgm_entry_size= -1`.

Step 3. Restart the driver

4.14.2 Flow Domains and Priorities

Flow steering defines the concept of domain and priority. Each domain represents a user agent that can attach a flow. The domains are prioritized. A higher priority domain will always supersede a lower priority domain when their flow specifications overlap. Setting a lower priority value will result in higher priority.

In addition to the domain, there is priority within each of the domains. Each domain can have at most 2^12 priorities in accordance with its needs.

The following are the domains at a descending order of priority:

- **User Verbs** allows a user application QP to be attached into a specified flow when using `ibv_exp_create_flow` and `ibv_exp_destroy_flow` verbs
  - `ibv_exp_create_flow`

```
struct ibv_exp_flow *ibv_exp_create_flow(struct ibv_qp *qp, struct ibv_exp_flow_attr
*flow)
```

Input parameters:

- `struct ibv_qp` - the attached QP.
- `struct ibv_exp_flow_attr` - attaches the QP to the flow specified. The flow contains mandatory control parameters and optional L2, L3 and L4 headers. The optional headers are detected by setting the size and `num_of_specs` fields:

  `struct ibv_exp_flow_attr` can be followed by the optional flow headers structs:

  ```
  struct ibv_exp_flow_spec_ib
  struct ibv_exp_flow_spec_eth
  struct ibv_exp_flow_spec_ipv4
  struct ibv_exp_flow_spec_tcp_udp
  ```

  For further information, please refer to the `ibv_exp_create_flow` man page.

Be advised that as of MLNX_OFED v2.0-3.0.0, the parameters (both the value and the mask) should be set in big-endian format.

Each header struct holds the relevant network layer parameters for matching. To enforce the match, the user sets a mask for each parameter. The supported masks are:

- All one mask - include the parameter value in the attached rule
  Note: Since the VLAN ID in the Ethernet header is 12bit long, the following parameter should be used: `flow_spec_eth.mask.vlan_tag = htons(0x0fff).`
- All zero mask - ignore the parameter value in the attached rule

When setting the flow type to NORMAL, the incoming traffic will be steered according to the rule specifications. ALL_DEFAULT and MC_DEFAULT rules options are valid only for Ethernet link type since InfiniBand link type packets always include QP number.

For further information, please refer to the relevant man pages.

- `ibv_exp_destroy_flow`

```
int ibv_exp_destroy_flow(struct ibv_exp_flow *flow_id)
```

Input parameters:

`ibv_exp_destroy_flow` requires `struct ibv_exp_flow` which is the return value of `ibv_exp_create_flow` in case of success.

Output parameters:

Returns 0 on success, or the value of errno on failure.

For further information, please refer to the `ibv_exp_destroy_flow` man page.

- **Ethtool**

Ethtool domain is used to attach an RX ring, specifically its QP to a specified flow.

Please refer to the most recent ethtool manpage for all the ways to specify a flow.

Examples:

- ethtool –U eth5 flow-type ether dst 00:11:22:33:44:55 loc 5 action 2

  All packets that contain the above destination MAC address are to be steered into rx-ring 2 (its underlying QP), with priority 5 (within the ethtool domain)
- ethtool –U eth5 flow-type tcp4 src-ip 1.2.3.4 dst-port 8888 loc 5 action 2

All packets that contain the above destination IP address and source port are to be steered into rx-ring 2. When destination MAC is not given, the user's destination MAC is filled automatically.

- ethtool -U eth5 flow-type ether dst 00:11:22:33:44:55 vlan 45 m 0xf000 loc 5 action 2

  All packets that contain the above destination MAC address and specific VLAN are steered into ring 2. Please pay attention to the VLAN's mask 0xf000. It is required in order to add such a rule.

- ethtool –u eth5

  Shows all of ethtool's steering rule

When configuring two rules with the same priority, the second rule will overwrite the first one, so this ethtool interface is effectively a table. Inserting Flow Steering rules in the kernel requires support from both the ethtool in the user space and in kernel (v2.6.28).

MLX4 Driver Support

The mlx4 driver supports only a subset of the flow specification the ethtool API defines. Asking for an unsupported flow specification will result with an "invalid value" failure.

The following are the flow specific parameters:

Table 7 - Flow Specific Parameters

	ether	tcp4/udp4	ip4
Mandatory	dst		src-ip/dst-ip
Optional	vlan	src-ip, dst-ip, src-port, dst-port, vlan	src-ip, dst-ip, vlan

- **RFS**

  RFS is an in-kernel-logic responsible for load balancing between CPUs by attaching flows to CPUs that are used by flow's owner applications. This domain allows the RFS mechanism to use the flow steering infrastructure to support the RFS logic by implementing the `ndo_rx_flow_steer`, which, in turn, calls the underlying flow steering mechanism with the RFS domain.

  Enabling the RFS requires enabling the `'ntuple'` flag via the ethtool,

  For example, to enable ntuple for eth0, run:

  ```
  ethtool -K eth0 ntuple on
  ```

  RFS requires the kernel to be compiled with the `CONFIG_RFS_ACCEL` option. This options is available in kernels 2.6.39 and above. Furthermore, RFS requires Device Managed Flow Steering support.

  RFS cannot function if LRO is enabled. LRO can be disabled via ethtool.

- **All of the rest**

  The lowest priority domain serves the following users:

  - **The mlx4 Ethernet driver** attaches its unicast and multicast MACs addresses to its QP using L2 flow specifications

- **The mlx4 ipoib driver** when it attaches its QP to his configured GIDS

Fragmented UDP traffic cannot be steered. It is treated as 'other' protocol by hardware (from the first packet) and not considered as UDP traffic.

We recommend using `libibverbs` v2.0-3.0.0 and `libmlx4` v2.0-3.0.0 and higher as of MLNX_OFED v2.0-3.0.0 due to API changes.

4.15 Single Root IO Virtualization (SR-IOV)

Single Root IO Virtualization (SR-IOV) is a technology that allows a physical PCIe device to present itself multiple times through the PCIe bus. This technology enables multiple virtual instances of the device with separate resources. Mellanox adapters are capable of exposing in ConnectX®-3 adapter cards up to 126 virtual instances called Virtual Functions (VFs). These virtual functions can then be provisioned separately. Each VF can be seen as an addition device connected to the Physical Function. It shares the same resources with the Physical Function, and its number of ports equals those of the Physical Function.

SR-IOV is commonly used in conjunction with an SR-IOV enabled hypervisor to provide virtual machines direct hardware access to network resources hence increasing its performance.

In this chapter we will demonstrate setup and configuration of SR-IOV in a Red Hat Linux environment using Mellanox ConnectX® VPI adapter cards family.

4.15.1 System Requirements

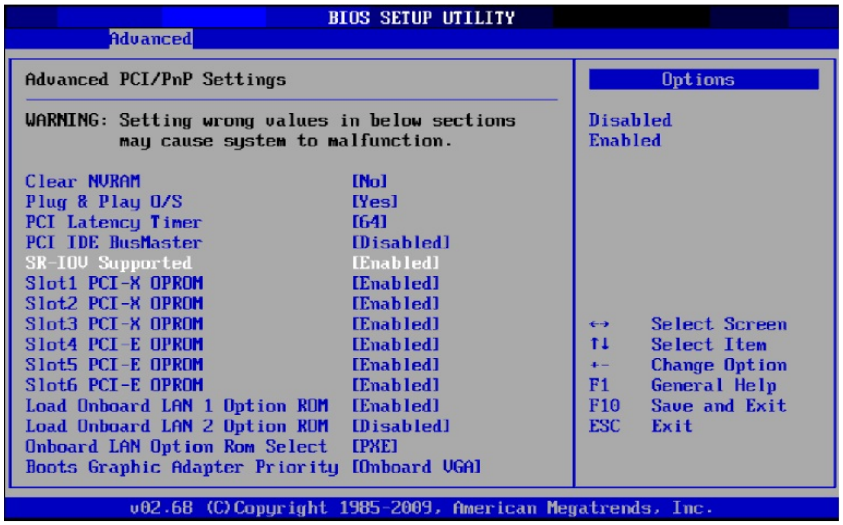
To set up an SR-IOV environment, the following is required:

- MLNX_OFED Driver
- A server/blade with an SR-IOV-capable motherboard BIOS
- Hypervisor that supports SR-IOV such as: Red Hat Enterprise Linux Server Version 6.*
- Mellanox ConnectX® VPI Adapter Card family with SR-IOV capability

4.15.2 Setting Up SR-IOV

Depending on your system, perform the steps below to set up your BIOS. The figures used in this section are for illustration purposes only. For further information, please refer to the appropriate BIOS User Manual:

Step 1. Enable "SR-IOV" in the system BIOS.


BIOS SETUP UTILITY
Advanced
Advanced PCI/PnP Settings
WARNING: Setting wrong values in below sections may cause system to malfunction.
Clear NVRAM [No]
Plug & Play O/S [Yes]
PCI Latency Timer [64]
PCI IDE BusMaster [Disabled]
SR-IOV Supported [Enabled]
Slot1 PCI-X OPROM [Enabled]
Slot2 PCI-X OPROM [Enabled]
Slot3 PCI-X OPROM [Enabled]
Slot4 PCI-E OPROM [Enabled]
Slot5 PCI-E OPROM [Enabled]
Slot6 PCI-E OPROM [Enabled]
Load Onboard LAN 1 Option ROM [Enabled]
Load Onboard LAN 2 Option ROM [Disabled]
Onboard LAN Option Rom Select [PXE]
Boots Graphic Adapter Priority [Onboard VGA]
Options
Disabled
Enabled
↔ Select Screen
↑↓ Select Item
+- Change Option
F1 General Help
F10 Save and Exit
ESC Exit
v02.68 (C)Copyright 1985-2009, American Megatrends, Inc.

Step 2. Enable "Intel Virtualization Technology".

BIOS SETUP UTILITY
Advanced
Microcode Rev :14
Cache L1 :256 KB
Cache L2 :1024 KB
Cache L3 :12288 KB
Ratio Status :Unlocked (Min:12, Max:18)
Ratio Actual Value:18
CPU Ratio [Auto]
C1E Support [Enabled]
Hardware Prefetcher [Enabled]
Adjacent Cache Line Prefetch [Enabled]
DCU Prefetcher [Enabled]
Data Reuse Optimization [Enabled]
MPS and ACPI MADT ordering [Modern ordering]
Intel(R) Virtualization Tech [Enabled]
Execute-Disable Bit Capability [Enabled]
Intel AES-NI [Disabled]
Simultaneous Multi-Threading [Enabled]
Active Processor Cores [All]
Intel(R) EIST Technology [Enabled]
When enabled, a VMM can utilize the additional HW Caps. provided by Intel(R) Virtualization Tech. Note: A full reset is required to change the setting.
↔ Select Screen
↑↓ Select Item
+- Change Option
F1 General Help
F10 Save and Exit
ESC Exit
v02.68 (C)Copyright 1985-2009, American Megatrends, Inc.

Step 3. Install a hypervisor that supports SR-IOV.

Step 4. Depending on your system, update the /boot/grub/grub.conf file to include a similar command line load parameter for the Linux kernel.

For example, to Intel systems, add:

```
default=0
timeout=5
splashimage=(hd0,0)/grub/splash.xpm.gz
hiddenmenu
title Red Hat Enterprise Linux Server (2.6.32-36.x86-645)
        root (hd0,0)
        kernel /vmlinuz-2.6.32-36.x86-64 ro root=/dev/VolGroup00/LogVol00 rhgb quiet
        intel_iommu=on[1]
        initrd /initrd-2.6.32-36.x86-64.img
```

1. Please make sure the parameter "intel_iommu=on" exists when updating the /boot/grub/grub.conf file, otherwise SR-IOV cannot be loaded.

Step 5. Install the MLNX_OFED driver for Linux that supports SR-IOV.

SR-IOV can be enabled and managed by using one of the following methods:

- Burn firmware with SR-IOV support where the number of virtual functions (VFs) will be set to 16

```
--enable-sriov
```

- Run the mlxconfig tool and set the `SRIOV_EN` parameter to "`1`" without re-burning the firmware

```
SRIOV_EN = 1
```

For further information, please refer to section *"mlxconfig - Changing Device Configuration Tool"* in the MFT User Manual (www.mellanox.com > Products > Software > Firmware Tools).

Step 6. Verify the HCA is configured to support SR-IOV.

```
[root@selene ~]# mstflint -dev <PCI Device> dc
```

1. Verify in the [HCA] section the following fields appear[1],[2]:

```
[HCA]
num_pfs = 1
total_vfs = <0-126>
sriov_en = true
```

Parameter	Recommended Value
num_pfs	1 **Note:** This field is optional and might not always appear.
total_vfs	• When using firmware version 2.31.5000 and above, the recommended value is 126. • When using firmware version 2.30.8000 and below, the recommended value is 63 **Note:** Before setting number of VFs in SR-IOV, please make sure your system can support that amount of VFs. Setting number of VFs larger than what your Hardware and Software can support may cause your system to cease working.
sriov_en	true

2. Add the above fields to the INI if they are missing.
3. Set the `total_vfs` parameter to the desired number if you need to change the number of total VFs.
4. Reburn the firmware using the mlxburn tool if the fields above were added to the INI, or the `total_vfs` parameter was modified.
   If the mlxburn is not installed, please downloaded it from the Mellanox website http://www.mellanox.com > products > Firmware tools

```
mlxburn -fw ./fw-ConnectX3-rel.mlx -dev /dev/mst/mt4099_pci_cr0 -conf ./MCX341A-XCG_Ax.ini
```

Step 7. Create the text file /etc/modprobe.d/mlx4_core.conf if it does not exist, otherwise delete its contents.

Step 8. Insert an "option" line in the /etc/modprobe.d/mlx4_core.conf file to set the number of VFs. the protocol type per port, and the allowed number of virtual functions to be used by the physical function driver (probe_vf).

For example:

```
options mlx4_core num_vfs=5 port_type_array=1,2 probe_vf=1
```

1. If SR-IOV is supported, to enable SR-IOV (if it is not enabled), it is sufficient to set “sriov_en = true” in the INI.
2. If the HCA does not support SR-IOV, please contact Mellanox Support: support@mellanox.com

Parameter	Recommended Value

num_vfs	• If absent, or zero: no VFs will be available
	• If its value is a single number in the range of 0-63: The driver will enable the `num_vfs` VFs on the HCA and this will be applied to all ConnectX® HCAs on the host.
	• If its a triplet x,y,z (applies only if all ports are configured as Ethernet) the driver creates:
	  • x single port VFs on physical port 1
	  • y single port VFs on physical port 2 (applies only if such a port exist)
	  • z n-port VFs (where n is the number of physical ports on device).
	  This applies to all ConnectX® HCAs on the host
	• If its format is a string: The string specifies the `num_vfs` parameter separately per installed HCA.
	  The string format is: "bb:dd.f-v,bb:dd.f-v,…"
	  • bb:dd.f = bus:device.function of the PF of the HCA
	  • v = number of VFs to enable for that HCA which is either a single value or a triplet, as described above.
	For example:
	• `num_vfs=5` - The driver will enable 5 VFs on the HCA and this will be applied to all ConnectX® HCAs on the host
	• `num_vfs=00:04.0-5,00:07.0-8` - The driver will enable 5 VFs on the HCA positioned in BDF 00:04.0 and 8 on the one in 00:07.0)
	• `num_vfs=1,2,3` - The driver will enable 1 VF on physical port 1, 2 VFs on physical port 2 and 3 dual port VFs (applies only to dual port HCA when all ports are Ethernet ports).
	• `num_vfs=00:04.0-5;6;7,00:07.0-8;9;10` - The driver will enable:
	  • HCA positioned in BDF 00:04.0
	  • 5 single VFs on port 1
	  • 6 single VFs on port 2
	  • 7 dual port VFs

	• HCA positioned in BDF 00:07.0
	  • 8 single VFs on port 1
	  • 9 single VFs on port 2
	  • 10 dual port VFs
	  Applies when all ports are configure as Ethernet in dual port HCAs
	Notes:
	• PFs not included in the above list will not have SR-IOV enabled.
	• Triplets and single port VFs are only valid when all ports are configured as Ethernet. When an InfiniBand port exists, only `num_vfs=a` syntax is valid where `"a"` is a single value that represents the number of VFs.
	• The second parameter in a triplet is valid only when there are more than 1 physical port.
	  In a triplet, `x+z<=63` and `y+z<=63`, the maximum number of VFs on each physical port must be 63.

Parameter	Recommended Value
port_type_array	Specifies the protocol type of the ports. It is either one array of 2 port types `'t1,t2'` for all devices or list of BDF to `port_type_array 'bb:dd.f-t1;t2,...'`. (string) Valid port types: 1-ib, 2-eth, 3-auto, 4-N/A If only a single port is available, use the N/A port type for port2 (e.g '1,4').

probe_vf	• If absent or zero: no VF interfaces will be loaded in the Hypervisor/host
	• If num_vfs is a number in the range of 1-63, the driver running on the Hypervisor will itself activate that number of VFs. All these VFs will run on the Hypervisor. This number will apply to all ConnectX® HCAs on that host.
	• If its a triplet x,y,z (applies only if all ports are configured as Ethernet), the driver probes:
	  • x single port VFs on physical port 1
	  • y single port VFs on physical port 2 (applies only if such a port exist)
	  • z n-port VFs (where n is the number of physical ports on device). Those VFs are attached to the hypervisor.
	• If its format is a string: the string specifies the `probe_vf` parameter separately per installed HCA.
	The string format is: "bb:dd.f-v,bb:dd.f-v,…
	  • bb:dd.f = bus:device.function of the PF of the HCA
	  • v = number of VFs to use in the PF driver for that HCA which is either a single value or a triplet, as described above

	For example: • `probe_vfs=5` - The PF driver will activate 5 VFs on the HCA and this will be applied to all ConnectX® HCAs on the host • `probe_vfs=00:04.0-5,00:07.0-8` - The PF driver will activate 5 VFs on the HCA positioned in BDF 00:04.0 and 8 for the one in 00:07.0)

Parameter	Recommended Value

	• `probe_vf=1,2,3` - The PF driver will activate 1 VF on physical port 1, 2 VFs on physical port 2 and 3 dual port VFs (applies only to dual port HCA when all ports are Ethernet ports).
	This applies to all ConnectX® HCAs in the host.
	• `probe_vf=00:04.0-5;6;7,00:07.0-8;9;10` - The PF driver will activate:
	  • HCA positioned in BDF 00:04.0
	  • 5 single VFs on port 1
	  • 6 single VFs on port 2
	  • 7 dual port VFs
	  • HCA positioned in BDF 00:07.0
	  • 8 single VFs on port 1
	  • 9 single VFs on port 2
	  • 10 dual port VFs
	Applies when all ports are configure as Ethernet in dual port HCAs.
	Notes:
	• PFs not included in the above list will not activate any of their VFs in the PF driver
	• Triplets and single port VFs are only valid when all ports are configured as Ethernet. When an InfiniBand port exist, only `probe_vf=a` syntax is valid where "`a`" is a single value that represents the number of VFs
	• The second parameter in a triplet is valid only when there are more than 1 physical port
	• Every value (either a value in a triplet or a single value) should be less than or equal to the respective value of `num_vfs` parameter

The example above loads the driver with 5 VFs (num_vfs). The standard use of a VF is a single VF per a single VM. However, the number of VFs varies upon the working mode requirements.

The protocol types are:

- Port 1 = IB
- Port 2 = Ethernet
  - port_type_array=2,2 (Ethernet, Ethernet)
  - port_type_array=1,1 (IB, IB)
  - port_type_array=1,2 (VPI: IB, Ethernet)
  - NO port_type_array module parameter: ports are IB

Step 9. Reboot the server.

If the SR-IOV is not supported by the server, the machine might not come out of boot/load.

Step 10. Load the driver and verify the SR-IOV is supported. Run:

```
lspci | grep Mellanox
03:00.0 InfiniBand: Mellanox Technologies MT26428 [ConnectX VPI PCIe 2.0 5GT/s - IB QDR
/ 10GigE] (rev b0)
03:00.1 InfiniBand: Mellanox Technologies MT27500 Family [ConnectX-3 Virtual Function]
(rev b0)
03:00.2 InfiniBand: Mellanox Technologies MT27500 Family [ConnectX-3 Virtual Function]
(rev b0)
03:00.3 InfiniBand: Mellanox Technologies MT27500 Family [ConnectX-3 Virtual Function]
(rev b0)
03:00.4 InfiniBand: Mellanox Technologies MT27500 Family [ConnectX-3 Virtual Function]
(rev b0)
03:00.5 InfiniBand: Mellanox Technologies MT27500 Family [ConnectX-3 Virtual Function]
(rev b0)
```

Where:

- “03:00" represents the Physical Function
- “03:00.**X**" represents the Virtual Function connected to the Physical Function

4.15.3 Enabling SR-IOV and Para Virtualization on the Same Setup

➢ ***To enable SR-IOV and Para Virtualization on the same setup:***

Step 1. Create a bridge.

```
vim /etc/sysconfig/network-scripts/ifcfg-bridge0
DEVICE=bridge0
TYPE=Bridge
IPADDR=12.195.15.1
NETMASK=255.255.0.0
BOOTPROTO=static
ONBOOT=yes
NM_CONTROLLED=no
DELAY=0
```

Step 2. Change the related interface (in the example below bridge0 is created over eth5).

```
DEVICE=eth5
BOOTPROTO=none
STARTMODE=on
HWADDR=00:02:c9:2e:66:52
TYPE=Ethernet
NM_CONTROLLED=no
ONBOOT=yes
BRIDGE=bridge0
```

Step 3. Restart the service network.

Step 4. Attach a virtual NIC to VM.

```
ifconfig -a
…
eth6      Link encap:Ethernet  HWaddr 52:54:00:E7:77:99
          inet addr:13.195.15.5  Bcast:13.195.255.255  Mask:255.255.0.0
          inet6 addr: fe80::5054:ff:fee7:7799/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
         RX packets:481 errors:0 dropped:0 overruns:0 frame:0
          TX packets:450 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:22440 (21.9 KiB)  TX bytes:19232 (18.7 KiB)
          Interrupt:10 Base address:0xa000
…
```

4.15.4 Assigning a Virtual Function to a Virtual Machine

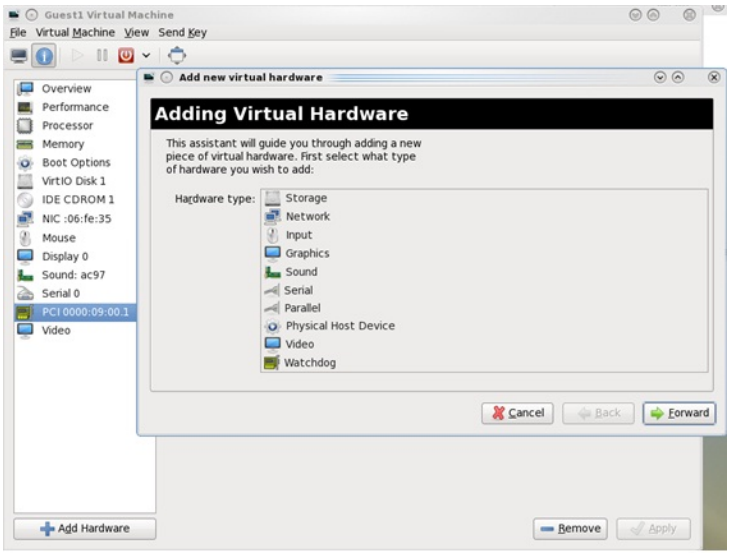
This section will describe a mechanism for adding a SR-IOV VF to a Virtual Machine.

4.15.4.1 Assigning the SR-IOV Virtual Function to the Red Hat KVM VM Server

Step 1. Run the virt-manager.

Step 2. Double click on the virtual machine and open its Properties.

Step 3. Go to Details->Add hardware ->PCI host device.



Step 4. Choose a Mellanox virtual function according to its PCI device (e.g., 00:03.1)

Step 5. If the Virtual Machine is up reboot it, otherwise start it.

Step 6. Log into the virtual machine and verify that it recognizes the Mellanox card. Run:

```
lspci | grep Mellanox

00:03.0 InfiniBand: Mellanox Technologies MT27500 Family [ConnectX-3 Virtual Function]
(rev b0)
```

Step 7. Add the device to the `/etc/sysconfig/network-scripts/ifcfg-ethX` configuration file. The MAC address for every virtual function is configured randomly, therefore it is not necessary to add it.

4.15.5 Uninstalling SR-IOV Driver

➢ ***To uninstall SR-IOV driver, perform the following:***

Step 1. For Hypervisors, detach all the Virtual Functions (VF) from all the Virtual Machines (VM) or stop the Virtual Machines that use the Virtual Functions.

Please be aware, stopping the driver when there are VMs that use the VFs, will cause machine to hang.

Step 2. Run the script below. Please be aware, uninstalling the driver deletes the entire driver's file, but does not unload the driver.

```
[root@swl022 ~]# /usr/sbin/ofed_uninstall.sh
This program will uninstall all OFED packages on your machine.
Do you want to continue?[y/N]:y
Running /usr/sbin/vendor_pre_uninstall.sh
Removing OFED Software installations
Running /bin/rpm -e --allmatches kernel-ib kernel-ib-devel libibverbs libibverbs-devel
libibverbs-devel-static libibverbs-utils libmlx4 libmlx4-devel libibcm libibcm-devel
libibumad libibumad-devel libibumad-static libibmad libibmad-devel libibmad-static
librdmacm librdmacm-utils librdmacm-devel ibacm opensm-libs opensm-devel perftest com-
pat-dapl compat-dapl-devel dapl dapl-devel dapl-devel-static dapl-utils srptools infini-
band-diags-guest ofed-scripts opensm-devel
warning: /etc/infiniband/openib.conf saved as /etc/infiniband/openib.conf.rpmsave
Running /tmp/2818-ofed_vendor_post_uninstall.sh
```

Step 3. Restart the server.

4.15.6 Configuring Pkeys and GUIDs under SR-IOV

4.15.6.1 Port Type Management

Port Type management is static when enabling SR-IOV (the `connectx_port_config` script will not work). The port type is set on the Host via a module parameter, `port_type_array`, in `mlx4_core`. This parameter may be used to set the port type uniformly for all installed ConnectX® HCAs, or it may specify an individual configuration for each HCA.

This parameter should be specified as an options line in the file `/etc/modprobe.d/mlx4_core.conf`.

For example, to configure all HCAs to have Port1 as IB and Port2 as ETH, insert the following line:

```
options mlx4_core port_type_array=1,2
```

To set HCAs individually, you may use a string of `Domain:bus:device.function=x;y`

For example, if you have a pair of HCAs, whose PFs are 0000:04:00.0 and 0000:05:00.0, you may specify that the first will have both ports as IB, and the second will have both ports as ETH as follows:

```
options mlx4_core port_type_array='0000:04:00.0-1;1,0000:05:00.0-2;2
```

Only the PFs are set via this mechanism. The VFs inherit their port types from their associated PF.

4.15.6.2 Virtual Function InfiniBand Ports

Each VF presents itself as an independent vHCA to the host, while a single HCA is observable by the network which is unaware of the vHCAs. No changes are required by the InfiniBand subsystem, ULPs, and applications to support SR-IOV, and vHCAs are interoperable with any existing (non-virtualized) IB deployments.

Sharing the same physical port(s) among multiple vHCAs is achieved as follows:

- Each vHCA port presents its own virtual GID table

  For further details, please refer to Section 4.15.6.2.3, on page 123.
- Each vHCA port presents its own virtual PKey table

  The virtual PKey table (presented to a VF) is a mapping of selected indexes of the physical PKey table. The host admin can control which PKey indexes are mapped to which virtual indexes using a sysfs interface. The physical PKey table may contain both full and partial memberships of the same PKey to allow different membership types in different virtual tables.
- Each vHCA port has its own virtual port state

  A vHCA port is up if the following conditions apply:
  - The physical port is up
  - The virtual GID table contains the GIDs requested by the host admin
  - The SM has acknowledged the requested GIDs since the last time that the physical port went up
- Other port attributes are shared, such as: GID prefix, LID, SM LID, LMC mask

To allow the host admin to control the virtual GID and PKey tables of vHCAs, a new sysfs 'iov' sub-tree has been added under the PF InfiniBand device.

4.15.6.2.1SR-IOV sysfs Administration Interfaces on the Hypervisor

Administration of GUIDs and PKeys is done via the sysfs interface in the Hypervisor (Dom0). This interface is under:

```
/sys/class/infiniband/<infiniband device>/iov
```

Under this directory, the following subdirectories can be found:

- `ports` - The actual (physical) port resource tables

  Port GID tables:
  - `ports/<n>/gids/<n> where 0 <= n <= 127` (the physical port gids)
  - `ports/<n>/admin_guids/<n> where 0 <= n <= 127` (allows examining or changing the administrative state of a given GUID>
  - `ports/<n>/pkeys/<n> where 0 <= n <= 126` (displays the contents of the physical pkey table)
- `<pci id> directories` - one for Dom0 and one per guest. Here, you may see the mapping between virtual and physical pkey indices, and the virtual to physical gid 0.

  Currently, the GID mapping cannot be modified, but the pkey virtual to physical mapping can .

  These directories have the structure:
  - `<pci_id>/port/<m>/gid_idx/0 where m = 1..2` (this is read-only)

    and

- <pci_id>/port/<m>/pkey_idx/<n>, where m = 1..2 and n = 0..126

For instructions on configuring pkey_idx, please see below.

4.15.6.2.2Configuring an Alias GUID (under ports/<n>/admin_guids)

Step 1. Determine the GUID index of the PCI Virtual Function that you want to pass through to a guest.

For example, if you want to pass through PCI function 02:00.3 to a certain guest, you initially need to see which GUID index is used for this function.

To do so:

```
cat /sys/class/infiniband/iov/0000:02:00.3/port/<port_num>/gid_idx/0
```

The value returned will present which guid index to modify on Dom0.

Step 2. Modify the physical GUID table via the admin_guids sysfs interface.

To configure the GUID at index <n> on port <port_num>:

```
cd /sys/class/infiniband/mlx4_0/iov/ports/<port_num>/admin_guids
echo <your desired guid>  > n
```

Example:

```
cd /sys/class/infiniband/mlx4_0/iov/ports/1/admin_guids
echo[1] "0x002fffff8118" > 3
```

1. echo "0x0" means let the SM assign a value to that GUID
   echo "0xffffffffffffffff" means delete that GUID
   echo <any other value> means request the SM to assign this GUID to this index

Step 3. Read the administrative status of the GUID index.

To read the administrative status of GUID index m on port n:

```
cat /sys/class/infiniband/mlx4_0/iov/ports/<n>/admin_guids/<m>
```

Step 4. Check the operational state of a GUID.

```
 /sys/class/infiniband/mlx4_0/iov/ports/<n>/gids  (where n = 1 or 2)
```

The values indicate what gids are actually configured on the firmware/hardware, and all the entries are R/O.

Step 5. Compare the value you read under the "admin_guids" directory at that index with the value under the "gids" directory, to verify the change requested in Step 3 has been accepted by the SM, and programmed into the hardware port GID table.

If the value under admin_guids/<m> is different that the value under gids/<m>, the request is still in progress.

4.15.6.2.3Multi-GUID Support in InfiniBand

As of MLNX_OFED v2.2-1.0.0, Infiniband VFs in SR-IOV setting can have more than a single GUID to be used for their purposes. Totally there are 128 GUIDs per port, where the PF occupies 2 entries, and the remaining GUIDs are divided equally between all the VFs. If there are any remainders, those GUIDs are given to the VFs with the lowest IDs.

For example in case that are 5 VFs, the division will be as follows: PF – 2, slave 1 will have 26 and last 4 slaves will have 25. (2 + 1 * 26 + 4 * 25 = 128).

To find the mapping between VF's GUID entry to its physical one, use the sysfs mechanism <gid_idx> directory.

The example below shows the mapping between "entry 0" of <PCI FUNCTION #1> to its physical one on port number 1.

```
cat /sys/class/infiniband/mlx4_0/iov/<PCI FUNCTION #1>/ports/1/gid_idx/0
```

Initial GUIDs' values depend on the mlx4_ib module parameter 'sm_guid_assign' as follows:

Mode Type	Description
sm assigned	Asks SM for values for GUID entry 0 per VF. Other entries will have value 0 in the port GUID table, and ffffffffffffffff under their matching admin_guids entry. This is the default mode (value=1). For example, as GUID entry 1 is not the base gid index of the HYP, upon startup it will have the below values under the sysfs entries: cat /sys/class/infiniband/mlx4_0/iov/ports/1/admin_guids/1 ffffffffffffffff cat /sys/class/infiniband/mlx4_0/iov/ports/1/gids/1 fe80:0000:0000:0000:0000:0000:0000:0000
admin assigned	Once sm_guid_assign is set to 0, the driver works in admin assigned mode which result in having only valid value for GUID entry 0 for the PF. All other entries under the port will be 0 and corresponding admin_guids will be ffffffffffffffff.

Port Up/Down

When moving to multi-guid mode, the port is assumed to be up when the base GUID index per VF/PF (entry 0) has a valid value. Setting other GUID entries does not affect the port status.

However, any change in a GUID will cause a GUID change event for its VF/PF even if it is not the base one.

Persistency Support

Once admin request is rejected by the SM, a retry mechanism is set. Retry time is set to 1 second, and for each retry it is multiplied by 2 until reaching the maximum value of 60 seconds. Additionally, when looking for the next record to be updated, the record having the lowest time to be executed is chosen.

Any value reset via the admin_guid interface is immediately executed and it resets this entry timer.

4.15.6.2.4Partitioning IPoIB Communication using PKeys

PKeys are used to partition IPoIB communication between the Virtual Machines and the Dom0 by mapping a non-default full-membership PKey to virtual index 0, and mapping the default PKey to a virtual pkey index other than zero.

The below describes how to set up two hosts, each with 2 Virtual Machines. Host-1/vm-1 will be able to communicate via IPoIB only with Host2/vm1,and Host1/vm2 only with Host2/vm2.

In addition, Host1/Dom0 will be able to communicate only with Host2/Dom0 over ib0. vm1 and vm2 will not be able to communicate with each other, nor with Dom0.

This is done by configuring the virtual-to-physical PKey mappings for all the VMs, such that at virtual PKey index 0, both vm-1s will have the same pkey and both vm-2s will have the same PKey (different from the vm-1's), and the Dom0's will have the default pkey (different from the vm's pkeys at index 0).

OpenSM must be used to configure the physical Pkey tables on both hosts.

- The physical Pkey table on both hosts (Dom0) will be configured by OpenSM to be:

```
index 0 = 0xffff
index 1 = 0xb000
index 2 = 0xb030
```

- The vm1's virt-to-physical PKey mapping will be:

```
pkey_idx 0 = 1
pkey_idx 1 = 0
```

- The vm2's virt-to-phys pkey mapping will be:

```
pkey_idx 0 = 2
pkey_idx 1 = 0
```

  so that the default pkey will reside on the vms at index 1 instead of at index 0.

The IPoIB QPs are created to use the PKey at index 0. As a result, the Dom0, vm1 and vm2 IPoIB QPs will all use different PKeys.

➢ ***To partition IPoIB communication using PKeys:***

Step 1. Create a file `"/etc/opensm/partitions.conf"` on the host on which OpenSM runs, containing lines.

```
Default=0x7fff,ipoib : ALL=full ;
Pkey1=0x3000,ipoib : ALL=full;
Pkey3=0x3030,ipoib : ALL=full;
```

This will cause OpenSM to configure the physical Port Pkey tables on all physical ports on the network as follows:

```
pkey idx | pkey value
---------|---------
       0 | 0xFFFF
       1 | 0xB000
       2 | 0xB030
```

(the most significant bit indicates if a PKey is a full PKey).

The `",ipoib"` causes OpenSM to pre-create IPoIB the broadcast group for the indicated PKeys.

Step 2. Configure (on Dom0) the virtual-to-physical PKey mappings for the VMs.

Step a. Check the PCI ID for the Physical Function and the Virtual Functions.

```
lspci | grep Mel
```

Step b. Assuming that on Host1, the physical function displayed by lspci is "0000:02:00.0", and that on Host2 it is "0000:03:00.0"
On Host1 do the following.

```
cd /sys/class/infiniband/mlx4_0/iov
0000:02:00.0  0000:02:00.1 0000:02:00.2 ...[1]
```

1. 0000:02:00.0 contains the virtual-to-physical mapping tables for the physical function.
   0000:02:00.X contain the virt-to-phys mapping tables for the virtual functions.

Do not touch the Dom0 mapping table (under <nnnn>:<nn>:00.0). Modify only tables under 0000:02:00.1 and/or 0000:02:00.2. We assume that vm1 uses VF 0000:02:00.1 and vm2 uses VF 0000:02:00.2

Step c. Configure the virtual-to-physical PKey mapping for the VMs.

```
echo 0 > 0000:02:00.1/ports/1/pkey_idx/1
echo 1 > 0000:02:00.1/ports/1/pkey_idx/0
echo 0 > 0000:02:00.2/ports/1/pkey_idx/1
echo 2 > 0000:02:00.2/ports/1/pkey_idx/0
```

vm1 pkey index 0 will be mapped to physical pkey-index 1, and vm2 pkey index 0 will be mapped to physical pkey index 2. Both vm1 and vm2 will have their pkey index 1 mapped to the default pkey.

Step d. On Host2 do the following.

```
cd /sys/class/infiniband/mlx4_0/iov
echo 0 > 0000:03:00.1/ports/1/pkey_idx/1
echo 1 > 0000:03:00.1/ports/1/pkey_idx/0
echo 0 > 0000:03:00.2/ports/1/pkey_idx/1
echo 2 > 0000:03:00.2/ports/1/pkey_idx/0
```

Step e. Once the VMs are running, you can check the VM's virtualized PKey table by doing (on the vm).

```
cat /sys/class/infiniband/mlx4_0/ports/[1,2]/pkeys/[0,1]
```

Step 3. Start up the VMs (and bind VFs to them).

Step 4. Configure IP addresses for ib0 on the host and on the guests.

4.15.6.3 Ethernet Virtual Function Configuration when Running SR-IOV

4.15.6.3.1VLAN Guest Tagging (VGT) and VLAN Switch Tagging (VST)

When running ETH ports on VGT, the ports may be configured to simply pass through packets as is from VFs (Vlan Guest Tagging), or the administrator may configure the Hypervisor to silently force packets to be associated with a VLan/Qos (Vlan Switch Tagging).

In the latter case, untagged or priority-tagged outgoing packets from the guest will have the VLAN tag inserted, and incoming packets will have the VLAN tag removed. Any vlan-tagged packets sent by the VF are silently dropped. The default behavior is VGT.

The feature may be controlled on the Hypervisor from userspace via iprout2 / netlink:

```
ip link set { dev DEVICE | group DEVGROUP } [ { up | down } ]
   ...
   [ vf NUM [ mac LLADDR ]
      [ vlan VLANID [ qos VLAN-QOS ] ]
           ...
   [ spoofchk { on | off} ] ]
           ...
```

use:

```
ip link set dev <PF device> vf <NUM> vlan <vlan_id> [qos <qos>]
```

- where `NUM = 0..max-vf-num`
- vlan_id = `0..4095` (4095 means "`set VGT`")
- qos = `0..7`

For example:

- `ip link set dev eth2 vf 2 qos 3` - sets VST mode for VF #2 belonging to PF eth2, with qos = 3
- `ip link set dev eth2 vf 2 4095` - sets mode for VF 2 back to VGT

4.15.6.3.2Additional Ethernet VF Configuration Options

- Guest MAC configuration

  By default, guest MAC addresses are configured to be all zeroes. In the mlnx_ofed guest driver, if a guest sees a zero MAC, it generates a random MAC address for itself. If the administrator wishes the guest to always start up with the same MAC, he/she should configure guest MACs before the guest driver comes up.

  The guest MAC may be configured by using:

  ```
  ip link set <PF device> vf <NUM> mac <LLADDR>
  ```

  For legacy guests, which do not generate random MACs, the adminstrator should always configure their MAC addresses via ip link, as above.

- Spoof checking

  Spoof checking is currently available only on upstream kernels newer than 3.1.

  ```
  ip link set dev <PF device> vf <NUM> spoofchk [on | off]
  ```

4.15.6.3.3Mapping VFs to Ports

➢ ***To view the VFs mapping to ports:***

Using the ip link tool v2.6.34~3 and above.

```
ip link
```

The output is as following:

```
61: p1p1: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT group
default qlen 1000
    link/ether 00:02:c9:f1:72:e0 brd ff:ff:ff:ff:ff:ff
    vf 0 MAC 00:00:00:00:00:00, vlan 4095, spoof checking off, link-state auto
    vf 37 MAC 00:00:00:00:00:00, vlan 4095, spoof checking off, link-state auto
    vf 38 MAC ff:ff:ff:ff:ff:ff, vlan 65535, spoof checking off, link-state disable
    vf 39 MAC ff:ff:ff:ff:ff:ff, vlan 65535, spoof checking off, link-state disable
```

When a MAC is ff:ff:ff:ff:ff:ff, the VF is not assigned to the port of the net device it is listed under. In the example above, **vf 38** is not assigned to the same port as **p1p1**, in contrast to **vf0**.

However, even VFs that are not assigned to the net device, could be used to set and change its settings. For example, the following is a valid command to change the spoof check:

```
ip link set dev p1p1 vf 38 spoofchk on
```

This command will affect only the **vf 38**. The changes can be seen in ip link on the net device that this device is assigned to.

4.15.6.3.4Mapping VFs to Ports using the mlnx_get_vfs.pl tool

- ***To map the PCI representation in BDF to the respective ports:***

```
mlnx_get_vfs.pl
```

The output is as following:

```
BDF 0000:04:00.0
        Port 1: 2
                vf0     0000:04:00.1
                vf1     0000:04:00.2
        Port 2: 2
                vf2     0000:04:00.3
                vf3     0000:04:00.4
        Both: 1
                vf4   0000:04:00.5
```

4.15.6.3.5RoCE Support

RoCE is supported on Virtual Functions and VLANs may be used with it. For RoCE, the hypervisor GID table size is of 16 entries while the VFs share the remaining 112 entries. When the number of VFs is larger than 56 entries, some of them will have GID table with only a single entry which is inadequate if VF's Ethernet device is assigned with an IP address.

When setting num_vfs in mlx4_core module parameter it is important to check that the number of the assigned IP addresses per VF does not exceed the limit for GID table size.

4.15.7 Running Network Diagnostic Tools on a Virtual Function

4.15.7.1 Overview

Until now, in MLNX_OFED, administrators were unable to run network diagnostics from a VF since sending and receiving Subnet Management Packets (SMPs) from a VF was not allowed, for security reasons: SMPs are not restricted by network partitioning and may affect the physical network topology. Moreover, even the SM may be denied access from portions of the network by setting management keys unknown to the SM.

However, it is desirable to grant SMP capability to certain privileged VFs, so certain network management activities may be conducted within virtual machines rather than only on the hypervisor.

4.15.7.2 Granting SMP Capability to a Virtual Function

To enable SMP capability for a VF, one must enable the Subnet Management Interface (SMI) for that VF. By default, the SMI interface is disabled for VFs. To enable SMI mads for VFs, there are two new sysfs entries per VF per on the Hypervisor (under `/sys/class/infiniband/mlx4_X/iov/<b.d.f>/ports/<1 or 2>`. These entries are displayed only for VFs (not for the PF), and only for IB ports (not ETH ports).

The first entry, `enable_smi_admin`, is used to enable SMI on a VF. By default, the value of this entry is zero (disabled). When set to "`1`", the SMI will be enabled for the VF on the next rebind or openibd restart on the VM that the VF is bound to. If the VF is currently bound, it must be unbound and then re-bound.

The second sysfs entry, `smi_enabled`, indicates the current enablement state of the SMI. 0 indicates disabled, and 1 indicates enabled. This entry is read-only.

When a VF is initialized (bound), during the initialization sequence, the driver copies the requested `smi_state` (`enable_smi_admin`) for that VF/port to the operational SMI state (`smi_enabled`) for that VF/port, and operate according to the operational state.

Thus, the sequence of operations on the hypevisor is:

Step 1. Enable SMI for any VF/port that you wish.

Step 2. Restart the VM that the VF is bound to (or just run `/etc/init.d/openibd restart` on that VM)

The SMI will be enabled for the VF/port combinations that you set in step 2 above. You will then be able to run network diagnostics from that VF.

4.15.7.3 Installing MLNX_OFED with Network Diagnostics on a VM

- ***To install mlnx_ofed on a VF which will be enabled to run the tools, run the following on the VM:***

```
# mlnx_ofed_install
```

4.16 Quantized Congestion Control

Congestion control is used to reduce packet drops in lossy environments and mitigate congestion spreading and resulting victim flows in lossless environments.

The Quantized Congestion Notification (QCN) IEEE standard (802.1Qau) provides congestion control for long-lived flows in limited bandwidth-delay product Ethernet networks. It is part of the IEEE Data Center Bridging (DCB) protocol suite, which also includes ETS, PFC, and DCBX. QCN in conducted at L2, and is targeted for hardware implementations. QCN applies to all Ethernet packets and all transports, and both the host and switch behavior is detailed in the standard.

QCN user interface allows the user to configure QCN activity. QCN configuration and retrieval of information is done by the `mlnx_qcn` tool. The command interface provides the user with a set of changeable attributes, and with information regarding QCN's counters and statistics. All parameters and statistics are defined per port and priority. QCN command interface is available if and only the hardware supports it.

4.16.1 QCN Tool - mlnx_qcn

mlnx_qcn is a tool used to configure QCN attributes of the local host. It communicates directly with the driver thus does not require setting up a DCBX daemon on the system.

The mlnx_qcn enables the user to:

- Inspect the current QCN configurations for a certain port sorted by priority
- Inspect the current QCN statistics and counters for a certain port sorted by priority
- Set values of chosen QCN parameters

Usage:

```
mlnx_qcn -i <interface> [options]
```

Options:

--version	Show program's version number and exit
-h, --help	Show this help message and exit
-i INTF, --interface=INTF	Interface name
-g TYPE, --get_type=TYPE	Type of information to get statistics/parameters
--rpg_enable=RPG_ENABLE_LIST	Set value of rpg_enable according to priority, use spaces between values and -1 for unknown values.
--rppp_max_rps=RPPP_MAX_RPS_LIST	Set value of rppp_max_rps according to priority, use spaces between values and -1 for unknown values.
--rpg_time_reset=RPG_TIME_RESET_LIST	Set value of rpg_time_reset according to priority, use spaces between values and -1 for unknown values.
--rpg_byte_reset=RPG_BYTE_RESET_LIST	Set value of rpg_byte_reset according to priority, use spaces between values and -1 for unknown values.
--rpg_threshold=RPG_THRESHOLD_LIST	Set value of rpg_threshold according to priority, use spaces between values and -1 for unknown values.
--rpg_max_rate=RPG_MAX_RATE_LIST	Set value of rpg_max_rate according to priority, use spaces between values and -1 for unknown values.
--rpg_ai_rate=RPG_AI_RATE_LIST	Set value of rpg_ai_rate according to priority, use spaces between values and -1 for unknown values.
--rpg_hai_rate=RPG_HAI_RATE_LIST	Set value of rpg_hai_rate according to priority, use spaces between values and -1 for unknown values.
--rpg_gd=RPG_GD_LIST	Set value of rpg_gd according to priority, use spaces between values and -1 for unknown values.

--rpg_min_dec_fac=RPG_MIN_DEC_FAC_LIST	Set value of rpg_min_dec_fac according to priority, use spaces between values and -1 for unknown values.
--rpg_min_rate=RPG_MIN_RATE_LIST	Set value of rpg_min_rate according to priority, use spaces between values and -1 for unknown values.
--cndd_state_machine=CNDD_STATE_MACHINE_LIST	Set value of cndd_state_machine according to priority, use spaces between values and -1 for unknown values.

- ***To get QCN current configuration sorted by priority:***

```
mlnx_qcn -i eth2 -g parameters
```

- ***To show QCN's statistics sorted by priority:***

```
mlnx_qcn -i eth2 -g statistics
```

Example output when running mlnx_qcn -i eth2 -g parameters:

```
priority 0:
        rpg_enable: 0
        rppp_max_rps: 1000
        rpg_time_reset: 1464
        rpg_byte_reset: 150000
        rpg_threshold: 5
        rpg_max_rate: 40000
        rpg_ai_rate: 10
        rpg_hai_rate: 50
        rpg_gd: 8
        rpg_min_dec_fac: 2
        rpg_min_rate: 10
        cndd_state_machine: 0
priority 1:
        rpg_enable: 0
        rppp_max_rps: 1000
        rpg_time_reset: 1464
        rpg_byte_reset: 150000
        rpg_threshold: 5
        rpg_max_rate: 40000
        rpg_ai_rate: 10
        rpg_hai_rate: 50
        rpg_gd: 8
        rpg_min_dec_fac: 2
        rpg_min_rate: 10
        cndd_state_machine: 0
.............................
.............................
priority 7:
        rpg_enable: 0
        rppp_max_rps: 1000
        rpg_time_reset: 1464
        rpg_byte_reset: 150000
        rpg_threshold: 5
```

```
rpg_max_rate: 40000
rpg_ai_rate: 10
rpg_hai_rate: 50
rpg_gd: 8
rpg_min_dec_fac: 2
rpg_min_rate: 10
cndd_state_machine: 0
```

4.16.2 Setting QCN Configuration

Setting the QCN parameters, requires updating its value for each priority. '-1' indicates no change in the current value.

Example for setting 'rp g_enable' in order to enable QCN for priorities 3, 5, 6:

```
mlnx_qcn -i eth2 --rpg_enable=-1 -1 -1 1 -1 1 1 -1
```

Example for setting 'rpg_hai_rate' for priorities 1, 6, 7:

```
mlnx_qcn -i eth2 --rpg_hai_rate=60 -1 -1 -1 -1 -1 60 60
```

4.17 CORE-Direct

4.17.1 CORE-Direct Overview

CORE-Direct provides a solution for off loading the MPI collectives operations from the software library to the network. CORE-Direct accelerates MPI applications and solves the scalability issues in large scale systems by eliminating the issues of operating systems noise and jitter.

It addresses the collectives communication scalability problem by off loading a sequence of data-dependent communications to the Host Channel Adapter (HCA). This solution provides the hooks needed to support computation and communication overlap. Additionally, it provides a means to reduce the effects of system noise and application skew on application scalability.

The relevant verbs to be used for CORE-Direct:

- ibv_exp_create_qp
- ibv_exp_modify_cq
- ibv_exp_query_device
- ibv_exp_post_task

Samples programs for reference:

- ibv_task_pingpong, ibv_cc_pingpong

4.18 Ethtool

ethtool is a standard Linux utility for controlling network drivers and hardware, particularly for wired Ethernet devices. It can be used to:

- Get identification and diagnostic information
- Get extended device statistics
- Control speed, duplex, autonegotiation and flow control for Ethernet devices
- Control checksum offload and other hardware offload features
- Control DMA ring sizes and interrupt moderation

The following are the ethtool supported options:

Table 8 - ethtool Supported Options

Options	Description

ethtool -i eth<x>	Checks driver and device information.
	For example:
	`#> ethtool -i eth2`
	`driver: mlx4_en (MT_0DD0120009_CX3)`
	`version: 2.1.6 (Aug 2013)`
	`firmware-version: 2.30.3000`
	`bus-info: 0000:1a:00.0`

ethtool -k eth<x>	Queries the stateless offload status.
ethtool -K eth<x> [rx on|off] [tx on|off] [sg on|off] [tso on|off] [lro on|off] [gro on|off] [gso on|off]	Sets the stateless offload status. TCP Segmentation Offload (TSO), Generic Segmentation Offload (GSO): increase outbound throughput by reducing CPU overhead. It works by queuing up large buffers and letting the network interface card split them into separate packets. Large Receive Offload (LRO): increases inbound throughput of high-bandwidth network connections by reducing CPU overhead. It works by aggregating multiple incoming packets from a single stream into a larger buffer before they are passed higher up the networking stack, thus reducing the number of packets that have to be processed. LRO is available in kernel versions < 3.1 for untagged traffic. **Note:** LRO will be done whenever possible. Otherwise GRO will be done. Generic Receive Offload (GRO) is available throughout all kernels.
ethtool -c eth<x>	Queries interrupt coalescing settings.
ethtool -C eth<x> adaptive-rx on|off	Enables/disables adaptive interrupt moderation. By default, the driver uses adaptive interrupt moderation for the receive path, which adjusts the moderation time to the traffic pattern.
ethtool -C eth<x> [pkt-rate-low N] [pkt-rate-high N] [rx-usecs-low N] [rx-usecs-high N]	Sets the values for packet rate limits and for moderation time high and low values. • Above an upper limit of packet rate, adaptive moderation will set the moderation time to its highest value. • Below a lower limit of packet rate, the moderation time will be set to its lowest value.

Table 8 - ethtool Supported Options

Options	Description
ethtool -C eth<x> [rx-usecs N] [rx-frames N]	Sets the interrupt coalescing settings when the adaptive moderation is disabled. **Note:** usec settings correspond to the time to wait after the *last* packet is sent/received before triggering an interrupt.
ethtool -a eth<x>	Queries the pause frame settings.
ethtool -A eth<x> [rx on|off] [tx on|off]	Sets the pause frame settings.
ethtool -g eth<x>	Queries the ring size values.
ethtool -G eth<x> [rx <N>] [tx <N>]	Modifies the rings size.
ethtool -S eth<x>	Obtains additional device statistics.
ethtool -t eth<x>	Performs a self diagnostics test.
ethtool -s eth<x> msglvl [N]	Changes the current driver message level.
ethtool -T eth<x>	Shows time stamping capabilities
ethtool -l eth<x>	Shows the number of channels
ethtool -L eth<x> [rx <N>] [tx <N>]	Sets the number of channels
ethtool --show-priv-flags eth<x>	Shows driver private flags and their states (on/off) Private flags are: • pm_qos_request_low_latency • mlx4_rss_xor_hash_function • qcn_disable_32_14_4_e
ethtool --set-priv-flags eth<x> <priv flag> <on/off>	Enables/disables driver feature matching the given private flag.

4.19 Dynamically Connected Transport Service

Dynamically Connected transport (DCT) service is an extension to transport services to enable a higher degree of scalability while maintaining high performance for sparse traffic. Utilization of DCT reduces the total number of QPs required system wide by having Reliable type QPs dynamically connect and disconnect from any remote node. DCT connections only stay connected while they are active. This results in smaller memory footprint, less overhead to set connections and higher on-chip cache utilization and hence increased performance. DCT is supported only in mlx5 driver.

4.20 PeerDirect

PeerDirect uses an API between IB CORE and peer memory clients, (e.g. GPU cards) to provide access to an HCA to read/write peer memory for data buffers. As a result, it allows RDMA-based (over InfiniBand/RoCE) application to use peer device computing power, and RDMA interconnect at the same time without copying the data between the P2P devices.

For example, PeerDirect is being used for GPUDirect RDMA.

Detailed description for that API exists under MLNX OFED installation, please see docs/readme_and_user_manual/PEER_MEMORY_API.txt

4.21 Inline-Receive

When Inline-Receive is active, the HCA may write received data in to the receive WQE or CQE. Using Inline-Receive saves PCIe read transaction since the HCA does not need to read the scatter list, therefore it improves performance in case of short receive-messages.

On poll CQ, the driver copies the received data from WQE/CQE to the user's buffers. Therefore, apart from querying Inline-Receive capability and Inline-Receive activation the feature is transparent to user application.

When Inline-Receive is active, user application must provide a valid virtual address for the receive buffers to allow the driver moving the inline-received message to these buffers. The validity of these addresses is not checked therefore the result of providing non-valid virtual addresses is unexpected.

Connect-IB™ supports Inline-Receive on both the requestor and the responder sides. Since data is copied at the poll CQ verb, Inline-Receive on the requestor side is possible only if the user chooses IB(V)_SIGNAL_ALL_WR.

4.21.1 Querying Inline-Receive Capability

User application can use the `ibv_exp_query_device` function to get the maximum possible Inline-Receive size. To get the size, the application needs to set the `IBV_EXP_DEVICE_ATTR_INLINE_RECV_SZ` bit in the `ibv_exp_device_attr comp_mask`.

4.21.2 Activating Inline-Receive

To activate the Inline-Receive, you need to set the required message size in the `max_inl_recv` field in the `ibv_exp_qp_init_attr struct` when calling `ibv_exp_create_qp` function. The value returned by the same field is the actual Inline-Receive size applied.

Setting the message size may affect the WQE/CQE size.

4.22 Ethernet Performance Counters

Counters are used to provide information about how well an operating system, an application, a service, or a driver is performing. The counter data helps determine system bottlenecks and fine-tune the system and application performance. The operating system, network, and devices provide counter data that an application can consume to provide users with a graphical view of how well the system is performing.

The counter index is a QP attribute given in the QP context. Multiple QPs may be associated with the same counter set, If multiple QPs share the same counter its value represents the cumulative total.

- ConnectX®-3 supports 127 different counters which are allocated as follow:
  - 4 counters reserved for PF - 2 counters for each port
  - 2 counters reserved for VF - 1 counter for each port
  - All other counters if exist are allocated by demand
- RoCE counters are available only through sysfs located under:
  - # /sys/class/infiniband/mlx4_*/ports/*/counters/
  - # /sys/class/infiniband/mlx4_*/ports/*/counters_ext/
- Physical Function can also read Virtual Functions' port counters through sysfs located under:
  - # /sys/class/net/eth*/vf*_statistics/

To display the network device Ethernet statistics, you can run:

```
Ethtool -S <devname>
```

Table 9 - Port IN Counters

Counter	Description
rx_packets	Total packets successfully received.
rx_bytes	Total bytes in successfully received packets.
rx_multicast_packets	Total multicast packets successfully received.
rx_broadcast_packets	Total broadcast packets successfully received.
rx_errors	Number of receive packets that contained errors preventing them from being deliverable to a higher-layer protocol.
rx_dropped	Number of receive packets which were chosen to be discarded even though no errors had been detected to prevent their being deliverable to a higher-layer protocol.
rx_length_errors	Number of received frames that were dropped due to an error in frame length
rx_over_errors	Number of received frames that were dropped due to overflow
rx_crc_errors	Number of received frames with a bad CRC that are not runts, jabbers, or alignment errors

Table 9 - Port IN Counters

Counter	Description
rx_jabbers	Number of received frames with a length greater than MTU octets and a bad CRC
rx_in_range_length_error	Number of received frames with a length/type field value in the (decimal) range [1500:46] (42 is also counted for VLANtagged frames)
rx_out_range_length_error	Number of received frames with a length/type field value in the (decimal) range [1535:1501]
rx_lt_64_bytes_packets	Number of received 64-or-less-octet frames
rx_127_bytes_packets	Number of received 65-to-127-octet frames
rx_255_bytes_packets	Number of received 128-to-255-octet frames
rx_511_bytes_packets	Number of received 256-to-511-octet frames
rx_1023_bytes_packets	Number of received 512-to-1023-octet frames
rx_1518_bytes_packets	Number of received 1024-to-1518-octet frames
rx_1522_bytes_packets	Number of received 1519-to-1522-octet frames
rx_1548_bytes_packets	Number of received 1523-to-1548-octet frames
rx_gt_1548_bytes_packets	Number of received 1549-or-greater-octet frames

Table 10 - Port OUT Counters

Counter	Description
tx_packets	Total packets successfully transmitted.
tx_bytes	Total bytes in successfully transmitted packets.
tx_multicast_packets	Total multicast packets successfully transmitted.
tx_broadcast_packets	Total broadcast packets successfully transmitted.
tx_errors	Number of frames that failed to transmit
tx_dropped	Number of transmitted frames that were dropped
tx_lt_64_bytes_packets	Number of transmitted 64-or-less-octet frames
tx_127_bytes_packets	Number of transmitted 65-to-127-octet frames
tx_255_bytes_packets	Number of transmitted 128-to-255-octet frames
tx_511_bytes_packets	Number of transmitted 256-to-511-octet frames
tx_1023_bytes_packets	Number of transmitted 512-to-1023-octet frames

Table 10 - Port OUT Counters

Counter	Description
tx_1518_bytes_packets	Number of transmitted 1024-to-1518-octet frames
tx_1522_bytes_packets	Number of transmitted 1519-to-1522-octet frames
tx_1548_bytes_packets	Number of transmitted 1523-to-1548-octet frames
tx_gt_1548_bytes_packets	Number of transmitted 1549-or-greater-octet frames

Table 11 - Port VLAN Priority Tagging (where <i> is in the range 0…7)

Counter	Description
rx_prio_<i>_packets	Total packets successfully received with priority i.
rx_prio_<i>_bytes	Total bytes in successfully received packets with priority i.
rx_novlan_packets	Total packets successfully received with no VLAN priority.
rx_novlan_bytes	Total bytes in successfully received packets with no VLAN priority.
tx_prio_<i>_packets	Total packets successfully transmitted with priority i.
tx_prio_<i>_bytes	Total bytes in successfully transmitted packets with priority i.
tx_novlan_packets	Total packets successfully transmitted with no VLAN priority.
tx_novlan_bytes	Total bytes in successfully transmitted packets with no VLAN priority.

Table 12 - Port Pause (where <i> is in the range 0…7)

Counter	Description
rx_pause_prio_<i>	The total number of PAUSE frames received from the far-end port
rx_pause_duration_prio_<i>	The total time in microseconds that far-end port was requested to pause transmission of packets.
rx_pause_transition_prio_<i>	The number of receiver transitions from XON state (paused) to XOFF state (non-paused)
tx_pause_prio_<i>	The total number of PAUSE frames sent to the far-end port
tx_pause_duration_prio_<i>	The total time in microseconds that transmission of packets has been paused

Table 12 - Port Pause (where <i> is in the range 0…7)

Counter	Description
tx_pause_transition_prio_<i>	The number of transmitter transitions from XON state (paused) to XOFF state (non-paused)

Table 13 - VPort Statistics (where <i>=<empty_string> is the PF, and ranges 1…NumOfVf per VF)

Counter	Description
vport<i>_rx_unicast_packets	Unicast packets received successfully
vport<i>_rx_unicast_bytes	Unicast packet bytes received successfully
vport<i>_rx_multicast_packets	Multicast packets received successfully
vport<i>_rx_multicast_bytes	Multicast packet bytes received successfully
vport<i>_rx_broadcast_packets	Broadcast packets received successfully
vport<i>_rx_broadcast_bytes	Broadcast packet bytes received successfully
vport<i>_rx_dropped	Received packets discarded due to out-of-buffer condition
vport<i>_rx_errors	Received packets discarded due to receive error condition
vport<i>_tx_unicast_packets	Unicast packets sent successfully
vport<i>_tx_unicast_bytes	Unicast packet bytes sent successfully
vport<i>_tx_multicast_packets	Multicast packets sent successfully
vport<i>_tx_multicast_bytes	Multicast packet bytes sent successfully
vport<i>_tx_broadcast_packets	Broadcast packets sent successfully
vport<i>_tx_broadcast_bytes	Broadcast packet bytes sent successfully
vport<i>_tx_errors	Packets dropped due to transmit errors

Table 14 - SW Statistics

Counter	Description
rx_lro_aggregated	Number of packets aggregated
rx_lro_flushed	Number of LRO flush to the stack
rx_lro_no_desc	Number of times LRO description was not found
rx_alloc_failed	Number of times failed preparing receive descriptor
rx_csum_good	Number of packets received with good checksum
rx_csum_none	Number of packets received with no checksum indication
tx_chksum_offload	Number of packets transmitted with checksum offload
tx_queue_stopped	Number of times transmit queue suspended
tx_wake_queue	Number of times transmit queue resumed
tx_timeout	Number of times transmitter timeout
tx_tso_packets	Number of packet that were aggregated

Table 15 - Per Ring (SW) Statistics (where <i> is the ring I – per configuration)

Counter	Description
rx<i>_packets	Total packets successfully received on ring i
rx<i>_bytes	Total bytes in successfully received packets on ring i.
tx<i>_packets	Total packets successfully transmitted on ring i.
tx<i>_bytes	Total bytes in successfully transmitted packets on ring i.

4.23 Memory Window

Memory Window allows the application to have a more flexible control over remote access to its memory. It is available only on physical functions / native machines The two types of Memory Windows supported are: type 1 and type 2B.

Memory Windows are intended for situations where the application wants to:

- grant and revoke remote access rights to a registered region in a dynamic fashion with less of a performance penalty
- grant different remote access rights to different remote agents and/or grant those rights over different ranges within registered region

For further information, please refer to the InfiniBand specification document.

Memory Windows API cannot co-work with peer memory clients (PeerDirect).

4.23.1 Query Capabilities

Memory Windows are available if and only the hardware supports it. To verify whether Memory Windows are available, run `ibv_exp_query_device`.

For example:

```
truct ibv_exp_device_attr device_attr = {.comp_mask = IBV_EXP_DEVICE_ATTR_RESERVED -
1};
ibv_exp_query_device(context, & device_attr);
if (device_attr.exp_device_cap_flags & IBV_EXP_DEVICE_MEM_WINDOW ||
                device_attr.exp_device_cap_flags & IBV_EXP_DEVICE_MW_TYPE_2B) {
/* Memory window is supported */
```

4.23.2 Allocating Memory Window

Allocating memory window is done by calling the `ibv_alloc_mw` verb.

```
type_mw = IBV_MW_TYPE_2/ IBV_MW_TYPE_1
mw = ibv_alloc_mw(pd, type_mw);
```

4.23.3 Binding Memory Windows

After allocated, memory window should be bound to a registered memory region. Memory Region should have been registered using the `IBV_EXP_ACCESS_MW_BIND` access flag.

- Binding Memory Window **type 1** is done via the `ibv_exp_bind_mw` verb.

```
struct ibv_exp_mw_bind mw_bind = { .comp_mask = IBV_EXP_BIND_MW_RESERVED - 1 };
ret = ibv_exp_bind_mw(qp, mw, &mw_bind);
```

- Binding memory window **type 2B** is done via the `ibv_exp_post_send` verb and a specific Work Request (WR) with `opcode = IBV_EXP_WR_BIND_MW`

Prior to binding, please make sure to update the existing rkey.

```
ibv_inc_rkey(mw->rkey)
```

4.23.4 Invalidating Memory Window

Before rebinding Memory Window type 2, it must be invalidated using the `ibv_exp_post_send` verb and a specific WR with `opcode = IBV_EXP_WR_LOCAL_INV`.

4.23.5 Deallocating Memory Window

Deallocating memory window is done using the `ibv_dealloc_mw` verb.

```
ibv_dealloc_mw(mw);
```

4.24 pm_qos usage on ingress Packet Traffic

pm_qos API is used by mlx4_en, to enforce minimum DMA latency requirement on the system when ingress traffic is detected. Additionally, it decreases packet loss on systems configured with abundant power state profile when Flow Control is disabled.

The pm_qos API contains the following functions:

- pm_qos_add_request
- pm_qos_update_request
- pm_qos_remove_request

pm_qos feature is both global and static, once a request is issued, it is enforced on all CPUs and does not change in time.

MLNX_OFED v2.2-1.0.0 provides an option to trigger a request when required and to remove it when no longer required. It is disabled by default and can be set/unset through the ethtool priv-flags.

For further information on how to enable/disable this feature, please refer to Table 8, “ethtool Supported Options,” on page 133.

4.25 XOR RSS Hash Function

The device has the ability to use XOR as the RSS distribution function, instead of the default Toplitz function.

The XOR function can be better distributed among driver's receive queues in small number of streams, where it distributes each TCP/UDP stream to a different queue.

MLNX_OFED v2.2-1.0.0 provides an option to change the working RSS hash function from Toplitz to XOR (and vice versa) through ethtool priv-flags.

For further information, please refer to Table 8, “ethtool Supported Options,” on page 133.

5 HPC Features

5.1 HPC-X

Mellanox HPC-X Scalable Software Toolkit provides various acceleration packages to improve both the performance and scalability of popular MPI and SHMEM/PGAS libraries. These packages, including MXM (Mellanox Messaging) which accelerates the underlying send/receive (or put/get) messages, and FCA (Fabric Collectives Accelerations) that accelerates the underlying collective operations used by the MPI/PGAS languages. This full-featured, tested and packaged version of HPC software enables MPI, SHMEM and PGAS programming languages to scale to extremely large clusters by improving on memory and latency related efficiencies, and to assure that the communication libraries are fully optimized of the Mellanox interconnect solutions.

The FCA and MXM software packages are also readily integrated into popular commercial and open-source MPI languages including OpenSHMEM / PGAS and the Berkeley UPC programming languages.

Mellanox OFED (MLNX_OFED) is a Mellanox tested and packaged version of the OpenFabrics Enterprise Distribution (OFED™) and supports RDMA (Remote DMA) and kernel bypass APIs called OFED verbs over InfiniBand and Ethernet.

Mellanox OFED and HPC-X allow OEM's and System Integrators to meet the needs of their endusers by deploying the latest available software that takes advantage of the features and capabilities available in the most recent hardware and firmware changes.

For further information, please refer to the HPC-X README file. (www.mellanox.com --> Products --> ScalableHPC --> HPC-X)

5.2 Shared Memory Access

The Shared Memory Access (SHMEM) routines provide low-latency, high-bandwidth communication for use in highly parallel scalable programs. The routines in the SHMEM Application Programming Interface (API) provide a programming model for exchanging data between cooperating parallel processes. The SHMEM API can be used either alone or in combination with MPI routines in the same parallel program.

The SHMEM parallel programming library is an easy-to-use programming model which uses highly efficient one-sided communication APIs to provide an intuitive global-view interface to shared or distributed memory systems. SHMEM's capabilities provide an excellent low level interface for PGAS applications.

A SHMEM program is of a single program, multiple data (SPMD) style. All the SHMEM processes, referred as processing elements (PEs), start simultaneously and run the same program. Commonly, the PEs perform computation on their own sub-domains of the larger problem, and periodically communicate with other PEs to exchange information on which the next communication phase depends.

The SHMEM routines minimize the overhead associated with data transfer requests, maximize bandwidth, and minimize data latency (the period of time that starts when a PE initiates a transfer of data and ends when a PE can use the data).

SHMEM routines support remote data transfer through:

- "put" operations - data transfer to a different PE

- "get" operations - data transfer from a different PE, and remote pointers, allowing direct references to data objects owned by another PE

Additional supported operations are collective broadcast and reduction, barrier synchronization, and atomic memory operations. An atomic memory operation is an atomic read-and-update operation, such as a fetch-and-increment, on a remote or local data object.

SHMEM libraries implement active messaging. The sending of data involves only one CPU where the source processor puts the data into the memory of the destination processor. Likewise, a processor can read data from another processor's memory without interrupting the remote CPU. The remote processor is unaware that its memory has been read or written unless the programmer implements a mechanism to accomplish this.

5.2.1 Mellanox ScalableSHMEM

The ScalableSHMEM programming library is a one-side communications library that supports a unique set of parallel programming features including point-to-point and collective routines, synchronizations, atomic operations, and a shared memory paradigm used between the processes of a parallel programming application.

Mellanox ScalableSHMEM is based on the API defined by the OpenSHMEM.org consortium. The library works with the OpenFabrics RDMA for Linux stack (OFED), and also has the ability to utilize MellanoX Messaging libraries (MXM) as well as Mellanox Fabric Collective Accelerations (FCA), providing an unprecedented level of scalability for SHMEM programs running over InfiniBand.

The latest ScalableSHMEM software can be downloaded from the Mellanox website.

5.2.2 Running SHMEM with FCA

The Mellanox Fabric Collective Accelerator (FCA) is a unique solution for offloading collective operations from the Message Passing Interface (MPI) or ScalableSHMEM process onto Mellanox InfiniBand managed switch CPUs. As a system-wide solution, FCA utilizes intelligence on Mellanox InfiniBand switches, Unified Fabric Manager and MPI nodes without requiring additional hardware. The FCA manager creates a topology based collective tree, and orchestrates an efficient collective operation using the switch-based CPUs on the MPI/ScalableSHMEM nodes.

FCA accelerates MPI/ScalableSHMEM collective operation performance by up to 100 times providing a reduction in the overall job runtime. Implementation is simple and transparent during the job runtime.

FCA is disabled by default and *must* be configured prior to using it from the ScalableSHMEM.

➢ ***To enable FCA by default in the ScalableSHMEM:***

1. Edit the `/opt/mellanox/openshmem/2.2/etc/openmpi-mca-params.conf` file.
2. Set the `scoll_fca_enable` parameter to `1`.

```
scoll_fca_enable=1
```

3. Set the `scoll_fca_np` parameter to 0.

```
scoll_fca_np=0
```

➢ ***To enable FCA in the*** `shmemrun` ***command line, add the following:***

```
-mca scoll_fca_enable=1
-mca scoll_fca_enable_np 0
```

➢ ***To disable FCA:***

```
-mca scoll_fca_enable 0 -mca coll_fca_enable 0
```

For more details on FCA installation and configuration, please refer to the FCA User Manual found in the Mellanox website.

5.2.3 Running ScalableSHMEM with MXM

MellanoX Messaging (MXM) library provides enhancements to parallel communication libraries by fully utilizing the underlying networking infrastructure provided by Mellanox HCA/switch hardware. This includes a variety of enhancements that take advantage of Mellanox networking hardware including:

- Multiple transport support including RC, XRC and UD
- Proper management of HCA resources and memory structures
- Efficient memory registration
- One-sided communication semantics
- Connection management
- Receive side tag matching
- Intra-node shared memory communication

These enhancements significantly increase the scalability and performance of message communi-cations in the network, alleviating bottlenecks within the parallel communication libraries

5.2.4 Running SHMEM with Contiguous Pages

Contiguous Pages improves performance by allocating user memory regions over contiguous pages. It enables a user application to ask low level drivers to allocate contiguous memory for it as part of `ibv_reg_mr`.

➢ ***To activate MLNX_OFED 2.0 and the contiguous pages allocator with SHMEM:***

Run the following argument to enable compound pages with SHMEM:

```
% /opt/mellanox/openshmem/2.1/bin/shmemrun -mca shmalloc_use_hugepages 5
```

If using compound pages is not possible, then the user will fall back to regular hugepages mechanism.

➢ ***To force use of compound pages allocator***

Run the following command:

```
% /opt/mellanox/openshmem/2.1/bin/shmemrun -mca shmalloc_use_hugepages 5 -x
MR_FORCE_CONTIG_PAGES=1
```

For further information on the Contiguous Pages, please refer to Section 4.11, “Contiguous Pages”, on page 105.

5.2.5 Running ScalableSHMEM Application

The ScalableSHMEM framework contains the `shmemrun` utility which launches the executable from a service node to compute nodes. This utility accepts the same command line parameters as `mpirun` from the OpenMPI package.

For further information, please refer to OpenMPI MCA parameters documentation at:

http://www.open-mpi.org/faq/?category=running.

Run `"shmemrun --help"` to obtain ScalableSHMEM job launcher runtime parameters.

ScalableSHMEM contains support for environment module system (http://modules.sf.net/). The modules configuration file can be found at:
`/opt/mellanox/openshmem/2.2/etc/shmem_modulefile`

5.3 Message Passing Interface

5.3.1 Overview

Mellanox OFED for Linux includes the following Message Passing Interface (MPI) implementations over InfiniBand:

- Open MPI 1.4.6 & 1.6.1 – an open source MPI-2 implementation by the Open MPI Project
- OSU MVAPICH2 1.7 – an MPI-1 implementation by Ohio State University

These MPI implementations, along with MPI benchmark tests such as OSU BW/LAT, Intel MPI Benchmark, and Presta, are installed on your machine as part of the Mellanox OFED for Linux installation. Table 16 lists some useful MPI links.

Table 16 - Useful MPI Links

MPI Standard	http://www-unix.mcs.anl.gov/mpi
Open MPI	http://www.open-mpi.org
MVAPICH 2 MPI	http://mvapich.cse.ohio-state.edu/
MPI Forum	http://www.mpi-forum.org

This chapter includes the following sections:

- Section 5.3.2,“Prerequisites for Running MPI,” on page 146
- Section 5.3.3,“MPI Selector - Which MPI Runs,” on page 148
- Section 5.3.4,“Compiling MPI Applications,” on page 148

5.3.2 Prerequisites for Running MPI

For launching multiple MPI processes on multiple remote machines, the MPI standard provides a launcher program that requires automatic login (i.e., password-less) onto the remote machines. SSH (Secure Shell) is both a computer program and a network protocol that can be used for logging and running commands on remote computers and/or servers.

5.3.2.1 SSH Configuration

The following steps describe how to configure password-less access over SSH:

Step 1. Generate an ssh key on the initiator machine (host1).

```
host1$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/<username>/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/<username>/.ssh/id_rsa.
Your public key has been saved in /home/<username>/.ssh/id_rsa.pub.
The key fingerprint is:
38:1b:29:df:4f:08:00:4a:0e:50:0f:05:44:e7:9f:05 <username>@host1
```

Step 2. Check that the public and private keys have been generated.

```
host1$ cd /home/<username>/.ssh/
host1$ ls
host1$ ls -la
total 40
drwx------  2 root root 4096 Mar  5 04:57 .
drwxr-x--- 13 root root 4096 Mar  4 18:27 ..
-rw-------  1 root root 1675 Mar  5 04:57 id_rsa
-rw-r--r--  1 root root  404 Mar  5 04:57 id_rsa.pub
```

Step 3. Check the public key.

```
host1$ cat id_rsa.pub
ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAQEA1zVY8VBHQh9okZN7OA1ibUQ74RXm4zHeczyVxpYHaDPyDmqezbYMKrCIVz
d10bH+ZkC0rpLYviU0oUHd3fvNTfMs0gcGg08PysUf+12FyYjira2P1xyg6mkHLGGqVutfEMmABZ3wNCUg6J2X
3G/uiuSWXeubZmbXcMrP/
w4IWByfH8ajwo6A5WioNbFZElbYeeNfPZf4UNcgMOAMWp64sL58tkt32F+RGmyLXQWZL27Synsn6dHpxMqBorX
NC0ZBe4kTnUqm63nQ2z1qVMdL9FrCma1xIOu9+SQJAjwONevaMzFKEHe7YHg6YrNfXunfdbEurzB524TpPcrod
ZlfCQ== <username>@host1
```

Step 4. Now you need to add the public key to the `authorized_keys2` file on the *target* machine.

```
host1$ cat id_rsa.pub | xargs ssh host2 \"echo  >>/home/<username>/.ssh/
authorized_keys2"
                                    <username>@host2's password:              // Enter
password
host1$
```

For a local machine, simply add the key to `authorized_keys2`.

```
host1$ cat id_rsa.pub >> authorized_keys2
```

Step 5. Test.

```
host1$ ssh  host2  uname
Linux
```

5.3.3 MPI Selector - Which MPI Runs

Mellanox OFED contains a simple mechanism for system administrators and end-users to select which MPI implementation they want to use. The MPI selector functionality is not specific to any MPI implementation; it can be used with any implementation that provides shell startup files that correctly set the environment for that MPI. The Mellanox OFED installer will automatically add MPI selector support for each MPI that it installs. Additional MPI's not known by the Mellanox OFED installer can be listed in the MPI selector; see the mpi-selector(1) man page for details.

Note that MPI selector only affects the default MPI environment for *future* shells. Specifically, if you use MPI selector to select MPI implementation ABC, this default selection will not take effect until you start a new shell (e.g., logout and login again). Other packages (such as environment modules) provide functionality that allows changing your environment to point to a new MPI implementation in the current shell. The MPI selector was not meant to duplicate or replace that functionality.

The MPI selector functionality can be invoked in one of two ways:

1. The mpi-selector-menu command.

   This command is a simple, menu-based program that allows the selection of the system-wide MPI (usually only settable by root) and a per-user MPI selection. It also shows what the current selections are. This command is recommended for all users.

2. The mpi-selector command.

   This command is a CLI-equivalent of the mpi-selector-menu, allowing for the same functionality as mpi-selector-menu but without the interactive menus and prompts. It is suitable for scripting.

5.3.4 Compiling MPI Applications

Compiling MVAPICH Applications

Please refer to http://mvapich.cse.ohio-state.edu/support/mvapich_user_guide.html.

To review the default configuration of the installation, check the default configuration file:

```
/usr/mpi/<compiler>/mvapich-<mvapich-ver>/etc/mvapich.conf
```

Compiling Open MPI Applications

Please refer to http://www.open-mpi.org/faq/?category=mpi-apps.

5.4 MellanoX Messaging

MellanoX Messaging (MXM) provides enhancements to parallel communication libraries by fully utilizing the underlying networking infrastructure provided by Mellanox HCA/switch hardware. This includes a variety of enhancements that take advantage of Mellanox networking hardware including:

- Multiple transport support including RC and UD
- Proper management of HCA resources and memory structures
- Efficient memory registration
- One-sided communication semantics

- Connection management
- Receive side tag matching
- Intra-node shared memory communication

These enhancements significantly increase the scalability and performance of message communications in the network, alleviating bottlenecks within the parallel communication libraries.

The latest MXM software can be downloaded from the Mellanox website.

5.4.1 Compiling Open MPI with MXM

Step 1. Install MXM from:

- an RPM

```
% rpm -ihv mxm-x.y.z-1.x86_64.rpm
```

- a tarball

```
% tar jxf mxm-x.y.z.tar.bz
```

MXM will be installed automatically in the `/opt/mellanox/mxm` folder.

Step 2. Enter Open MPI source directory and run:

```
% cd $OMPI_HOME
% ./configure --with-mxm=/opt/mellanox/mxm <... other configure parameters...>
% make all && make install
```

Older versions of MLNX_OFED come with pre-installed older MXM and Open MPI versions. Please uninstall any old MXM version prior to installing the latest MXM version in order to use it with older MLNX_OFED versions.

Table 17 - MLNX_OFED and MXM Versions

MLNX_OFED Version	MXM Version
v1.5.3-3.1.0 and v2.0-3.0.0	MXM v1.x and Open MPI compiled with MXM v1.x
v2.0-3.0.0 and higher	MXM v2.x/3.x and Open MPI compiled with MXM v2.x/3.x

To check the version of MXM installed on your host, run:

```
% rpm -qi mxm
```

➢ ***To upgrade MLNX_OFED v1.5.3-3.1.0 or later with a newer MXM:***

Step 1. Remove MXM.

```
# rpm -e mxm
```

Step 2. Remove the pre-compiled Open MPI.

```
# rpm -e mlnx-openmpi_gcc
```

Step 3. Install the new MXM and compile the Open MPI with it.

To run Open MPI without MXM, run:

```
% mpirun -mca mtl ^mxm <...>
```

When upgrading to MXM v0.52, Open MPI compiled with the previous versions of the MXM should be recompiled with MXM v0.52.

5.4.2 Enabling MXM in OpenMPI

As of MXM v2.1, MXM is automatically selected by Open MPI (up to v1.6) when the Number of Processes (NP) is higher or equal to 128.

➢ ***To activate MXM for any NP, run:***

```
% mpirun -mca mtl_mxm_np 0 <...other mpirun parameters ...>
```

From Open MPI v1.7.x, MXM is selected when the number of processes is higher or equal to 0. i.e. by default.

5.4.3 Tuning MXM Settings

The default MXM settings are already optimized. To check the available MXM parameters and their default values, run the `/opt/mellanox/mxm/bin/mxm_dump_config -f` utility which is part of the MXM RPM.

MXM parameters can be modified in one of the following methods:

- Modifying the default MXM parameters value as part of the mpirun:

```
% mpirun -x MXM_UD_RX_MAX_BUFFS=128000 <...>
```

- Modifying the default MXM parameters value from SHELL:

```
% export MXM_UD_RX_MAX_BUFFS=128000
% mpirun <...>
```

5.4.4 Configuring Multi-Rail Support

Multi-Rail support enables the user to use more than one of the active ports on the card, by making a better use of the resources. It provides a combined throughput among the used ports.

Multi-Rail support in MXM v0.52 allows different processes on the same host to use different active ports. Every process can only use one port (as opposed to MXM v1.5).

➢ ***To configure dual rail support:***

- Specify the list of ports you would like to use to enable multi rail support.

```
-x MXM_RDMA_PORTS=cardName:portNum
```

or

```
-x MXM_IB_PORTS=cardName:portNum
```

For example:

```
-x MXM_IB_PORTS=mlx5_0:1
```

It is also possible to use several HCAs and ports during the run (separated by a comma):

```
-x MXM_IB_PORTS=mlx5_0:1,mlx5_1:1
```

MXM will bind a process to one of the HCA ports from the given ports list according to the MXM_IB_MAP_MODE parameter (for load balancing).

Possible values for MXM_IB_MAP_MODE are:

- first - [Default] Maps the first suitable HCA port to all processes
- affinity - Distributes the HCA ports evenly among processes based on CPU affinity
- nearest - Tries to find the nearest HCA port based on CPU affinity

You may also use an asterisk (`*`) and a question mark (`?`) to choose the HCA and the port you would like to use.

- * - use all active cards/ports that are available
- ? - use the first active card/port that is available

  For example:

```
-x MXM_IB_PORTS=*:?
```

  will take all the active HCAs and the first active port on each of them.

5.4.5 Configuring MXM over the Ethernet Fabric

➢ ***To configure MXM over the Ethernet fabric:***

Step 1. Make sure the Ethernet port is active.

```
% ibv_devinfo
```

ibv_devinfo displays the list of cards and ports in the system. Please make sure (in the ibv_devinfo output) that the desired port has Ethernet at the `link_layer` field and that its state is `PORT_ACTIVE`.

Step 2. Specify the ports you would like to use, if there is a non Ethernet active port in the card.

```
-x MXM_RDMA_PORTS=mlx4_0:1
```

or

```
-x MXM_IB_PORTS=mlx4_0:1
```

5.4.6 Configuring MXM over Different Transports

MXM v3.0 supports the following transports.

- Intra node communication via Shared Memory with KNEM support
- Unreliable Datagram (UD)
- Reliable Connected (RC)
- SELF transport - a single process communicates with itself
- Dynamically Connected Transport (DC) (at beta level)

  Note: DC is supported on Connect-IB® HCAs with MLNX_OFED v2.1-1.0.0 and higher.

  To use DC set the following:

  - in the command line:

```
% mpirun -x MXM_TLS=self,shm,dc
```

  - from the SHELL:

```
% export MXM_TLS=self,shm,dc
```

By default the transports (TLS) used are: `MXM_TLS=self,shm,ud`

5.4.7 Configuring Service Level Support

Service Level Support is currently at alpha level.
Please be aware that the content below is subject to change.

MXM v3.0 added support for Service Level to enable Quality of Service (QoS). If set, every InfiniBand endpoint in MXM will generate a random Service Level (SL) within the given range, and use it for outbound communication.

Setting the value is done via the following environment parameter:

```
MXM_IB_NUM_SLS
```

Available Service Level values are 1-16 where the default is 1.

5.5 Fabric Collective Accelerator

To meet the needs of scientific research and engineering simulations, supercomputers are growing at an unrelenting rate. As supercomputers increase in size from mere thousands to hundreds-of-thousands of processor cores, new performance and scalability challenges have emerged. In the past, performance tuning of parallel applications could be accomplished fairly easily by separately optimizing their algorithms, communication, and computational aspects. However, as systems continue to scale to larger machines, these issues become co-mingled and must be addressed comprehensively.

Collective communications execute global communication operations to couple all processes/nodes in the system and therefore must be executed as quickly and as efficiently as possible. Indeed, the scalability of most scientific and engineering applications is bound by the scalability and performance of the collective routines employed. Most current implementations of collective operations will suffer from the effects of systems noise at extreme-scale (system noise increases the latency of collective operations by amplifying the effect of small, randomly occurring OS interrupts during collective progression.) Furthermore, collective operations will consume a significant fraction of CPU cycles, cycles that could be better spent doing meaningful computation.

Mellanox Technologies has addressed these two issues, lost CPU cycles and performance lost to the effects of system noise, by offloading the communications to the host channel adapters (HCAs) and switches. The technology, named CORE-Direct® (Collectives Offload Resource Engine), provides the most advanced solution available for handling collective operations thereby ensuring maximal scalability, minimal CPU overhead, and providing the capability to overlap communication operations with computation allowing applications to maximize asynchronous communication.

Users may benefit immediately from CORE-Direct® out-of-the-box by simply specifying the necessary BCOL/SBGP combinations. In order to take maximum advantage of CORE-Direct®, users may modify their applications to use MPI 3.0 non-blocking routines while using CORE-Direct® to offload the collective "under-the-covers", thereby allowing maximum opportunity to overlap communication with computation.

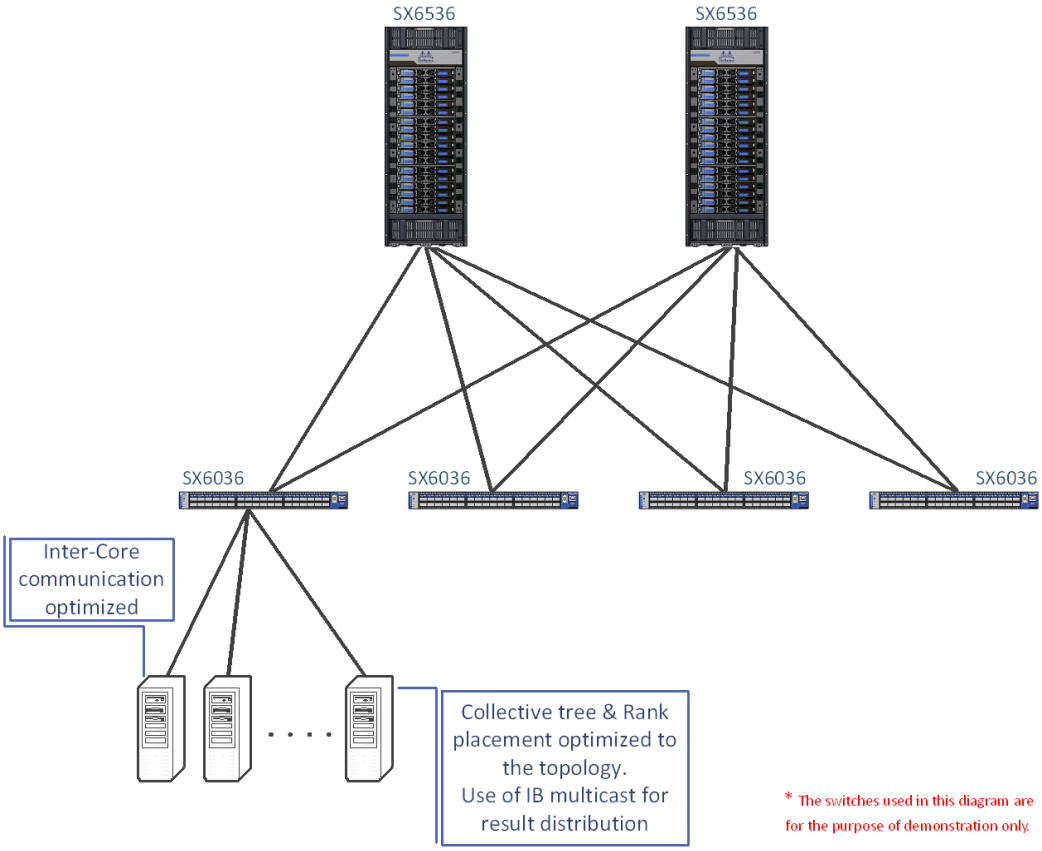
Additionally, FCA 3.0 also contains support to build runtime configurable hierarchical collectives. We currently support socket and UMA level discovery with network topology slated for future versions. As with FCA 2.X we also provide the ability to accelerate collectives with hard-

ware multicast. In FCA 3.0 we also expose the performance and scalability of Mellanox's advanced point-to-point library, MXM 2.x, in the form of the "mlnx_p2p" BCOL. This allows users to take full advantage of new features with minimal effort.

FCA 3.0 is a standalone library that can be integrated into any MPI or PGAS runtime. Support for FCA 3.0 is currently integrated into Open MPI versions 1.7.4 and higher. The 3.0 release currently supports blocking and non-blocking variants of "Allgather", "Allreduce", "Barrier", and "Bcast".

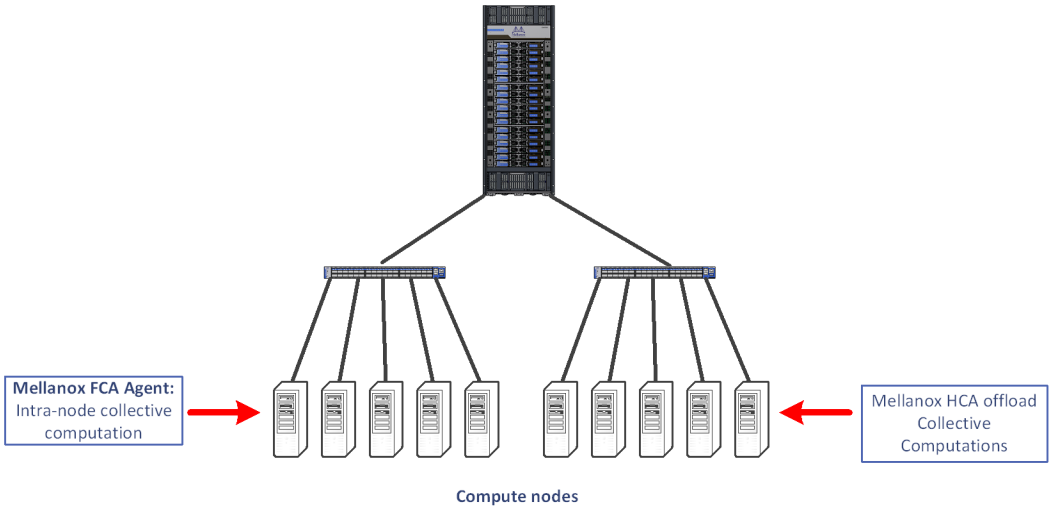
The following diagram summarizes the FCA architecture:

Figure 4: FCA Architecture



The following diagram shows the FCA components and the role that each plays in the acceleration process:

Figure 5: FCA Components



After MLNX_OFED installation, FCA can be found at `/opt/mellanox/fca` folder.

For further information on configuration instructions, please refer to the FCA User Manual.

5.6 ScalableUPC

Unified Parallel C (UPC) is an extension of the C programming language designed for high performance computing on large-scale parallel machines.The language provides a uniform programming model for both shared and distributed memory hardware. The programmer is presented with a single shared, partitioned address space, where variables may be directly read and written by any processor, but each variable is physically associated with a single processor. UPC uses a Single Program Multiple Data (SPMD) model of computation in which the amount of parallelism is fixed at program startup time, typically with a single thread of execution per processor.

In order to express parallelism, UPC extends ISO C 99 with the following constructs:

- An explicitly parallel execution model
- A shared address space
- Synchronization primitives and a memory consistency model
- Memory management primitives

The UPC language evolved from experiences with three other earlier languages that proposed parallel extensions to ISO C 99: AC, Split-C, and Parallel C Preprocessor (PCP). UPC is not a superset of these three languages, but rather an attempt to distill the best characteristics of each. UPC combines the programmability advantages of the shared memory programming paradigm and the control over data layout and performance of the message passing programming paradigm.

Mellanox ScalableUPC is based on Berkely UPC package (see http://upc.lbl.gov/) and contains the following enhancements:

- GasNet library used within UPC integrated with Mellanox FCA which off-loads from UPC collective operations.
  For further information on FCA, please refer to the Mellanox website.

- GasNet library contains MXM conduit which offloads from UPC all P2P operations as well as some synchronization routines. For further information on MXM, please refer to the Mellanox website.

Mellanox OFED 1.8 includes ScalableUPC 2.1, which is installed under: `/opt/mellanox/bupc`.
If you have installed OFED 1.8, you do *not* need to download and install ScalableUPC.

Mellanox ScalableUPC is distributed as source RPM as well and can be downloaded from the Mellanox website.

5.6.1 Installing ScalableUPC

Mellanox ScalableUPC is installed as part of MLNX_OFED package.

Mellanox OFED 1.8.5 includes ScalableUPC Rev 2.2, which is installed under: `/opt/mellanox/bupc`.
If you have installed OFED 1.8.5, you do not need to download and install ScalableUPC.

Mellanox ScalableUPC is distributed as source RPM as well and can be downloaded from the Mellanox website.

Please note, the binary distribution of ScalableUPC is compiled with the following defaults:

- FCA support. FCA is disabled at runtime by default and must be configured prior to using it from the ScalableUPC. For further information, please refer to FCA User Manual.
- MXM support enabled by default

5.6.2 FCA Runtime Parameters

The following parameters can be passed to "`upcrun`" in order to change FCA support behavior:

Table 18 - Runtime Parameters

Parameter	Description
`-fca_enable <0|1>`	Disables/Enables FCA support at runtime (default: disable).
`-fca_np <value>`	Enables FCA support for collective operations if the number of processes in the job is greater than the `fca_np` value (default: 64).
`-fca_verbose <level>`	Sets verbosity level for the FCA modules

Table 18 - Runtime Parameters

Parameter	Description
`-fca_ops <+/->[op_list]`	`op_list` - comma separated list of collective operations. • `-fca_ops <+/->[op_list]` - Enables/disables only the specified operations • `-fca_ops <+/->` - Enables/disables all operations By default all operations are enabled. Allowed operation names are: barrier (br), bcast (bt), reduce (rc), allgather (ag). Each operation can be also enabled/disabled via environment variable: • GASNET_FCA_ENABLE_BARRIER • GASNET_FCA_ENABLE_BCAST, • GASNET_FCA_ENABLE_REDUCE, **Note:** All the operations are enabled by default.

5.6.2.1 Enabling FCA Operations through Environment Variables in ScalableUPC

This method can be used to control UPC FCA offload from environment using job scheduler srun utility. The valid values are: 1 - enable, 0 - disable.

➢ ***To enable a specific operation with shell environment variables in ScalableUPC:***

```
% export GASNET_FCA_ENABLE_BARRIER=1
% export GASNET_FCA_ENABLE_BCAST=1
% export GASNET_FCA_ENABLE_REDUCE=1
```

5.6.2.2 Controlling FCA Offload in ScalableUPC using Environment Variables

➢ ***To enable FCA module under ScalableUPC:***

```
% export GASNET_FCA_ENABLE_CMD_LINE=1
```

➢ ***To set FCA verbose level:***

```
% export GASNET_FCA_VERBOSE_CMD_LINE=10
```

➢ ***To set the minimal number of processes threshold to activate FCA:***

```
% export GASNET_FCA_NP_CMD_LINE=1
```

ScalableUPC contains modules configuration file (http://modules.sf.net) which can be found at `/opt/mellanox/bupc/2.2/etc/bupc_modulefile`.

5.6.3 Various Executable Examples

The following are various executable examples.

➢ ***To run a ScalableUPC application without FCA support:***

```
% upcrun -np 128 -fca_enable 0 <executable filename>
```

➢ ***To run UPC applications with FCA enabled for any number of processes:***

```
% export GASNET_FCA_ENABLE_CMD_LINE=1 GASNET_FCA_NP_CMD_LINE=0
% upcrun -np 64 <executable filename>
```

- ***To run UPC application on 128 processes, verbose mode:***

```
% upcrun -np 128 -fca_enable 1 -fca_np 10 -fca_verbose 5 <executable filename>
```

- ***To run UPC application, offload to FCA Barrier and Broadcast only:***

```
% upcrun -np 128 -fca_ops +barrier,bt <executable filename>
```

6 Working With VPI

VPI allows ConnectX ports to be independently configured as either IB or Eth.

6.1 Port Type Management

ConnectX ports can be individually configured to work as InfiniBand or Ethernet ports. By default both ConnectX ports are initialized as InfiniBand ports. If you wish to change the port type use the `connectx_port_config` script after the driver is loaded.

Running "`/sbin/connectx_port_config -s`" will show current port configuration for all ConnectX devices.

Port configuration is saved in the file: `/etc/infiniband/connectx.conf`. This saved configuration is restored at driver restart only if restarting via "`/etc/init.d/openibd restart`".

Possible port types are:

- eth – Ethernet
- ib – Infiniband
- auto – Link sensing mode - Detect port type based on the attached network type. If no link is detected, the driver retries link sensing every few seconds.

The port link type can be configured for each device in the system at run time using the "`/sbin/connectx_port_config`" script. This utility will prompt for the PCI device to be modified (if there is only one it will be selected automatically).

In the next stage the user will be prompted for the desired mode for each port. The desired port configuration will then be set for the selected device.

This utility also has a non-interactive mode:

```
/sbin/connectx_port_config [[-d|--device <PCI device ID>] -c|--conf <port1,port2>]"
```

6.2 Auto Sensing

Auto Sensing enables the NIC to automatically sense the link type (InfiniBand or Ethernet) based on the link partner and load the appropriate driver stack (InfiniBand or Ethernet).

For example, if the first port is connected to an InfiniBand switch and the second to Ethernet switch, the NIC will automatically load the first switch as InfiniBand and the second as Ethernet.

6.2.1 Enabling Auto Sensing

Upon driver start up:

1. Sense the adapter card's port type:

   If a valid cable or module is connected (QSFP, SFP+, or SFP with EEPROM in the cable/module):

   - Set the port type to the sensed link type (IB/Ethernet)

   Otherwise:

   - Set the port type as default (Ethernet)

During driver run time:

- Sense a link every 3 seconds if no link is sensed/detected
- If sensed, set the port type as sensed

7 Performance

For further information on Linux performance, please refer to the Performance Tuning Guide for Mellanox Network Adapters.

8 OpenSM – Subnet Manager

8.1 Overview

OpenSM is an InfiniBand compliant Subnet Manager (SM). It is provided as a fixed flow executable called *opensm*, accompanied by a testing application called *osmtest*. OpenSM implements an InfiniBand compliant SM according to the InfiniBand Architecture Specification chapters: Management Model (13), Subnet Management (14), and Subnet Administration (15).

8.2 opensm Description

opensm is an InfiniBand compliant Subnet Manager and Subnet Administrator that runs on top of the Mellanox OFED stack. **opensm** performs the InfiniBand specification's required tasks for initializing InfiniBand hardware. One SM must be running for each InfiniBand subnet.

opensm also provides an experimental version of a performance manager.

opensm defaults were designed to meet the common case usage on clusters with up to a few hundred nodes. Thus, in this default mode, **opensm** will scan the IB fabric, initialize it, and sweep occasionally for changes.

opensm attaches to a specific IB port on the local machine and configures only the fabric connected to it. (If the local machine has other IB ports, **opensm** will ignore the fabrics connected to those other ports). If no port is specified, **opensm** will select the first "best" available port. **opensm** can also present the available ports and prompt for a port number to attach to.

By default, the **opensm** run is logged to two files: `/var/log/messages` and `/var/log/opensm.log`. The first file will register only general major events, whereas the second file will include details of reported errors. All errors reported in this second file should be treated as indicators of IB fabric health issues. (Note that when a fatal and non-recoverable error occurs, **opensm** will exit). Both log files should include the message "SUBNET UP" if **opensm** was able to setup the subnet correctly.

8.2.1 opensm Syntax

```
opensm [OPTIONS]
```

where OPTIONS are:

```
--version
          Prints OpenSM version and exits.

--config, -F <file-name>
          The name of the OpenSM config file. When not specified
          /etc/opensm/opensm.conf will be used (if exists).

--create-config, -c <file-name>
          OpenSM will dump its configuration to the specified file and exit.
          This is a way to generate OpenSM configuration file template.

--guid, -g <GUID in hex>
          This option specifies the local port GUID value
          with which OpenSM should bind.  OpenSM may be
```

```
          bound to 1 port at a time.
          If GUID given is 0, OpenSM displays a list
          of possible port GUIDs and waits for user input.
          Without -g, OpenSM tries to use the default port.

--lmc, -l <LMC>
          This option specifies the subnet's LMC value.
          The number of LIDs assigned to each port is 2^LMC.
          The LMC value must be in the range 0-7.
          LMC values > 0 allow multiple paths between ports.
          LMC values > 0 should only be used if the subnet
          topology actually provides multiple paths between
          ports, i.e. multiple interconnects between switches.
          Without -l, OpenSM defaults to LMC = 0, which allows
          one path between any two ports.

--priority, -p <PRIORITY>
          This option specifies the SM's PRIORITY.
          This will effect the handover cases, where master
          is chosen by priority and GUID.  Range goes
          from 0 (lowest priority) to 15 (highest).

--smkey, -k <SM_Key>
          This option specifies the SM's SM_Key (64 bits).
          This will effect SM authentication.
          Note that OpenSM version 3.2.1 and below used the
          default value '1' in a host byte order, it is fixed
          now but you may need this option to interoperate
          with old OpenSM running on a little endian machine.

--reassign_lids, -r
          This option causes OpenSM to reassign LIDs to all
          end nodes. Specifying -r on a running subnet
          may disrupt subnet traffic.
          Without -r, OpenSM attempts to preserve existing
          LID assignments resolving multiple use of same LID.

--routing_engine, -R <engine name>
          This option chooses routing engine(s) to use instead of default
          Min Hop algorithm.  Multiple routing engines can be specified
          separated by commas so that specific ordering of routing
          algorithms will be tried if earlier routing engines fail.
          If all configured routing engines fail, OpenSM will always
          attempt to route with Min Hop unless 'no_fallback' is
          included in the list of routing engines.
          Supported engines: updn, file, ftree, lash, dor, torus-2QoS, chain
```

```
--do_mesh_analysis
          This option enables additional analysis for the lash
          routing engine to precondition switch port assignments
          in regular cartesian meshes which may reduce the number
          of SLs required to give a deadlock free routing

--lash_start_vl <vl number>
          Sets the starting VL to use for the lash routing algorithm.
          Defaults to 0.
--sm_sl <sl number>
          Sets the SL to use to communicate with the SM/SA. Defaults to 0.

--connect_roots, -z
          This option enforces routing engines (up/down and
          fat-tree) to make connectivity between root switches
          and in this way be IBA compliant. In many cases,
          this can violate "pure" deadlock free algorithm, so
          use it carefully.

--ucast_cache, -A
          This option enables unicast routing cache to prevent
          routing recalculation (which is a heavy task in a
          large cluster) when there was no topology change
          detected during the heavy sweep, or when the topology
          change does not require new routing calculation,
          e.g. in case of host reboot.
          This option becomes very handy when the cluster size
          is thousands of nodes.

--lid_matrix_file, -M <file name>
          This option specifies the name of the lid matrix dump file
          from where switch lid matrices (min hops tables will be
          loaded.

--lfts_file, -U <file name>
          This option specifies the name of the LFTs file
          from where switch forwarding tables will be loaded.

--sadb_file, -S <file name>
          This option specifies the name of the SA DB dump file
          from where SA database will be loaded.

--root_guid_file, -a <path to file>
          Set the root nodes for the Up/Down or Fat-Tree routing
          algorithm to the guids provided in the given file (one
```

```
          to a line)

--cn_guid_file, -u <path to file>
          Set the compute nodes for the Fat-Tree routing algorithm
          to the guids provided in the given file (one to a line)

--io_guid_file, -G <path to file>
          Set the I/O nodes for the Fat-Tree routing algorithm
          to the guids provided in the given file (one to a line)

--port-shifting
          Attempt to shift port routes around to remove alignment problems
          in routing tables

--scatter-ports <random seed>
          Randomize best port chosen for a route

--max_reverse_hops, -H <hop_count>
          Set the max number of hops the wrong way around
          an I/O node is allowed to do (connectivity for I/O nodes on top swithces)

--ids_guid_file, -m <path to file>
          Name of the map file with set of the IDs which will be used
          by Up/Down routing algorithm instead of node GUIDs
          (format: <guid> <id> per line)

--guid_routing_order_file, -X <path to file>
          Set the order port guids will be routed for the MinHop
          and Up/Down routing algorithms to the guids provided in the
          given file (one to a line)

--torus_config <path to file>
          This option defines the file name for the extra configuration
          info needed for the torus-2QoS routing engine.   The default
          name is '/etc/opensm/torus-2QoS.conf'

--once, -o
          This option causes OpenSM to configure the subnet
          once, then exit.  Ports remain in the ACTIVE state.

--sweep, -s <interval>
          This option specifies the number of seconds between
          subnet sweeps.  Specifying -s 0 disables sweeping.
          Without -s, OpenSM defaults to a sweep interval of
          10 seconds.
```

```
--timeout, -t <milliseconds>
          This option specifies the time in milliseconds
          used for transaction timeouts.
          Specifying -t 0 disables timeouts.
          Without -t, OpenSM defaults to a timeout value of
          200 milliseconds.

--retries <number>
          This option specifies the number of retries used
          for transactions.
          Without --retries, OpenSM defaults to 3 retries
          for transactions.

--maxsmps, -n <number>
          This option specifies the number of VL15 SMP MADs
          allowed on the wire at any one time.
          Specifying --maxsmps 0 allows unlimited outstanding
          SMPs.
          Without --maxsmps, OpenSM defaults to a maximum of
          4 outstanding SMPs.

--rereg_on_guid_migr
          This option if enabled, forces OpenSM to send port info
          with client reregister bit set to all nodes in the
          fabric when alias Guid migrates from one physical port
          to another.

--aguid_inout_notice
          This option enables sending GID IN/OUT notices on Alias GUIDs
          register/delete request to registered clients.

--sm_assign_guid_func [uniq_count | base_port]
          Specifies the algorithm that SM will use when it comes to choose
          SM assigned alias GUIDs. The default is uniq_count.

--console, -q [off|local]
          This option activates the OpenSM console (default off).

--ignore-guids, -i <equalize-ignore-guids-file>
          This option provides the means to define a set of ports
          (by guid) that will be ignored by the link load
          equalization algorithm.

--hop_weights_file, -w <path to file>
          This option provides the means to define a weighting
          factor per port for customizing the least weight
          hops for the routing.
```

```
--port_search_ordering_file, -O <path to file>
          This option provides the means to define a mapping
          between ports and dimension (Order) for controlling
          Dimension Order Routing (DOR).
          Moreover this option provides the means to define non
          default routing port order.

--dimn_ports_file, -O <path to file> (DEPRECATED)
          This option provides the means to define a mapping
          between ports and dimension (Order) for controlling
          Dimension Order Routing (DOR).

--honor_guid2lid, -x
          This option forces OpenSM to honor the guid2lid file,
          when it comes out of Standby state, if such file exists
          under OSM_CACHE_DIR, and is valid. By default, this is FALSE.

--const_multicast
          This option forces OpenSM to conserver previously built
          multicast trees

--log_file, -f <log-file-name>
          This option defines the log to be the given file.
          By default, the log goes to /var/log/opensm.log.
          For the log to go to standard output use -f stdout.

--log_limit, -L <size in MB>
          This option defines maximal log file size in MB. When
          specified the log file will be truncated upon reaching
          this limit.

--erase_log_file, -e
          This option will cause deletion of the log file
          (if it previously exists). By default, the log file
          is accumulative.

--Pconfig, -P <partition-config-file>
          This option defines the optional partition configuration file.
          The default name is '/etc/opensm/partitions.conf'.

--no_part_enforce, -N (DEPRECATED)
          This option disables partition enforcement on switch external ports.
```

```
--part_enforce, -Z [both, in, out, off]
          This option indicates the partition enforcement type (for switches)
          Enforcement type can be outbound only (out), inbound only (in), both or
          disabled (off). Default is both.

--allow_both_pkeys, -W
          This option indicates whether both full and limited membership
          on the same partition can be configured in the PKeyTable.
          Default is not to allow both pkeys.

--qos, -Q
          This option enables QoS setup.

--qos_policy_file, -Y <QoS-policy-file>
          This option defines the optional QoS policy file.
          The default name is '/etc/opensm/qos-policy.conf'.

--congestion_control
          (EXPERIMENTAL) This option enables congestion control configuration.

--cc_key <key>
          (EXPERIMENTAL) This option configures the CCkey to use when configuring
          congestion control.

--stay_on_fatal, -y
          This option will cause SM not to exit on fatal initialization
          issues: if SM discovers duplicated guids or 12x link with
          lane reversal badly configured.
          By default, the SM will exit on these errors.

--daemon, -B
          Run in daemon mode - OpenSM will run in the background.

--inactive, -I
           Start SM in inactive rather than normal init SM state.

--perfmgr
           Start with PerfMgr enabled.

--perfmgr_sweep_time_s <sec.>
           PerfMgr sweep interval in seconds.

--prefix_routes_file <path to file>
          This option specifies the prefix routes file.
          Prefix routes control how the SA responds to path record
          queries for off-subnet DGIDs.  Default file is:
              /etc/opensm/prefix-routes.conf
```

```
--consolidate_ipv6_snm_req
          Use shared MLID for IPv6 Solicited Node Multicast groups
          per MGID scope and P_Key.

--consolidate_ipv4_mask
          Use mask for IPv4 multicast groups multiplexing
          per MGID scope and P_Key.

--pid_file <path to file>
          Specifies the file that contains the process ID of the
          opensm daemon.The default is /var/run/opensm.pid
--max_seq_redisc
          Specifies the maximum number of failed discovery loops
          done by the SM, before completing the whole heavy sweep cycle

--mc_secondary_root_guid <GUID in hex>,
          This option defines the guid of the multicast secondary root switch

--mc_primary_root_guid <GUID in hex>,
          This option defines the guid of the multicast primary root switch

--guid_routing_order_no_scatter
           Don't use scatter for ports defined in guid_routing_order file

--sa_pr_full_world_queries_allowed
          This option allows OpenSM to respond full World Path Record queries
          (path record for each pair of ports in a fabric).

--enable_crashd
          This option causes OpenSM to run Crash Daemon child process that allows
          backtrace dump in case of fatal terminating signals.

--log_prefix <prefix text>
          Prefix to syslog messages from OpenSM.

--verbose, -v
          This option increases the log verbosity level.
          The -v option may be specified multiple times
          to further increase the verbosity level.
          See the -D option for more information about
          log verbosity.

--V, -V
          This option sets the maximum verbosity level and
          forces log flushing.
          The -V is equivalent to '-D 0xFF -d 2'.
          See the -D option for more information about
          log verbosity.

--D, -D <flags>
```

```
          This option sets the log verbosity level.
          A flags field must follow the -D option.
          A bit set/clear in the flags enables/disables a
          specific log level as follows:
          BIT    LOG LEVEL ENABLED
          ----   ----------------
          0x01 - ERROR (error messages)
          0x02 - INFO (basic messages, low volume)
          0x04 - VERBOSE (interesting stuff, moderate volume)
          0x08 - DEBUG (diagnostic, high volume)
          0x10 - FUNCS (function entry/exit, very high volume)
          0x20 - FRAMES (dumps all SMP and GMP frames)
          0x40 - ROUTING (dump FDB routing information)
          0x80 - currently unused.
          Without -D, OpenSM defaults to ERROR + INFO (0x3).
          Specifying -D 0 disables all messages.
          Specifying -D 0xFF enables all messages (see -V).
          High verbosity levels may require increasing
          the transaction timeout with the -t option.

--debug, -d <number>
          This option specifies a debug option.
          These options are not normally needed.
          The number following -d selects the debug
          option to enable as follows:
          OPT   Description
          ---    ----------------
          -d0  - Ignore other SM nodes
          -d1  - Force single threaded dispatching
          -d2  - Force log flushing after each log message
          -d3  - Disable multicast support
          -d10 - Put OpenSM in testability mode
          Without -d, no debug options are enabled

--help, -h, -?
          Display this usage info then exit.
```

8.2.2 Environment Variables

The following environment variables control `opensm` behavior:

- OSM_TMP_DIR

  Controls the directory in which the temporary files generated by opensm are created. These files are: `opensm-subnet.lst`, `opensm.fdbs`, and `opensm.mcfdbs`. By default, this directory is `/var/log`.

- OSM_CACHE_DIR

`opensm` stores certain data to the disk such that subsequent runs are consistent. The default directory used is `/var/cache/opensm`. The following file is included in it:

- `guid2lid` – stores the LID range assigned to each GUID

8.2.3 Signaling

When OpenSM receives a HUP signal, it starts a new heavy sweep as if a trap has been received or a topology change has been found.

Also, SIGUSR1 can be used to trigger a reopen of `/var/log/opensm.log` for logrotate purposes.

8.2.4 Running opensm

The defaults of **opensm** were designed to meet the common case usage on clusters with up to a few hundred nodes. Thus, in this default mode, **opensm** will scan the IB fabric, initialize it, and sweep occasionally for changes.

To run **opensm** in the default mode, simply enter:

```
host1# opensm
```

Note that **opensm** needs to be run on at least one machine in an IB subnet.

By default, an **opensm** run is logged to two files: `/var/log/messages` and `/var/log/opensm.log`. The first file, `message`, registers only general major events; the second file, `opensm.log`, includes details of reported errors. All errors reported in `opensm.log` should be treated as indicators of IB fabric health. Both log files should include the message "SUBNET UP" if opensm was able to setup the subnet correctly.

If a fatal, non-recoverable error occurs, opensm exits.

8.2.4.1 Running OpenSM As Daemon

OpenSM can also run as daemon. To run OpenSM in this mode, enter:

```
host1# /etc/init.d/opensmd start
```

8.3 osmtest Description

osmtest is a test program for validating the InfiniBand Subnet Manager and Subnet Administrator. **osmtest** provides a test suite for **opensm**. It can create an inventory file of all available nodes, ports, and PathRecords, including all their fields. It can also verify the existing inventory with all the object fields, and matches it to a pre-saved one. See Section 8.3.2.

osmtest has the following test flows:

- Multicast Compliancy test
- Event Forwarding test
- Service Record registration test
- RMPP stress test
- Small SA Queries stress test

8.3.1 Syntax

```
osmtest [OPTIONS]
```

where OPTIONS are:

```
-f, --flow       This option directs osmtest to run a specific flow:
                 Flow  Description:
                 c = create an inventory file with all nodes, ports and
                  paths
                 a = run all validation tests (expecting an input
                  inventory)
                 v = only validate the given inventory file
                 s = run service registration, deregistration, and lease
                  test
                 e = run event forwarding test
                 f = flood the SA with queries according to the stress mode
                 m = multicast flow
                 q = QoS info: dump VLArb and SLtoVL tables
                 t = run trap 64/65 flow (this flow requires running of
                  external tool)
                    Default = all flows except QoS
-w, --wait       This option specifies the wait time for trap 64/65 in
               seconds. It is used only when running -f t - the trap 64/
               65 flow Default = 10 sec
-d, --debug      This  option  specifies  a  debug option.  These options
              are not normally needed.  The number  following  -d
              selects  the  debug option to enable as follows:
                OPT   Description
                ---   -----------------
                -d0   Ignore other SM nodes
                -d1   Force single threaded dispatching
                -d2   Force log flushing after each log message
                -d3   Disable multicast support
-m, --max_lid    This  option specifies the maximal LID number to be
              searched for during inventory file build (Default = 100)
-g, --guid       This option specifies the  local  port  GUID  value  with
              which OpenSM  should  bind.   OpenSM may be bound to
               1 port at a time. If GUID given is 0, OpenSM displays  a
                 list  of  possible  port GUIDs  and waits for user input.
              Without -g, OpenSM tries to use the default port.
-p, --port       This option displays a menu of possible local port  GUID
              values with which osmtest could bind
-i, --inventory   This  option  specifies the name of the inventory file
               Normally, osmtest expects to find an inventory file,
               which osmtest uses to validate real-time information
               received from the SA during testing. If -i  is  not
```

```
                specified,  osmtest  defaults  to  the  file
                osmtest.dat.See -c option for related information
-s, --stress      This option runs the specified stress test instead of the
               normal test suite Stress test options are as follows:
                  OPT    Description
                  ---    -----------------
                  -s1    Single-MAD response SA queries
                  -s2    Multi-MAD (RMPP) response SA queries
                  -s3    Multi-MAD (RMPP) Path Record SA queries
                  Without -s, stress testing is not performed
-M, --Multicast_ModeThis option specify length of Multicast test:
                  OPT    Description
                  ---    -----------------
                  -M1    Short Multicast Flow (default) - single mode
                  -M2    Short Multicast Flow - multiple mode
                  -M3    Long Multicast Flow - single mode
                  -M4    Long Multicast Flow - multiple mode
                Single mode - Osmtest is tested alone, with no other  apps
                  that interact with OpenSM MC
                  Multiple  mode  -  Could  be  run  with other apps using
               MC with OpenSM. Without -M, default flow testing is per
               formed
-t, --timeout     This option specifies the time in milliseconds used for
               transaction  timeouts.  Specifying -t 0 disables
               timeouts. Without -t, OpenSM defaults to a timeout value
               of 200 milliseconds.
-l, --log_file    This option defines the log to be the given  file.   By
                default the log goes to /var/log/osm.log.  For the log to
               go to standard output use -f stdout.
-v, --verbose     This option increases the log verbosity level.   The  -v
               option may  be  specified  multiple  times to further
               increase the verbosity level.  See the -vf option for
               more  information  about log verbosity.
-V                This  option  sets  the  maximum  verbosity level and
               forces log flushing.  The -V is equivalent to '-vf 0xFF -
               d 2'.  See the -vf option for more information about log
               verbosity.
-vf               This  option  sets  the log verbosity level.  A flags
               field must follow the -D option.  A bit set/clear in the
               flags enables/disables a specific log level as follows:
                  BIT    LOG LEVEL ENABLED
                  ----   -----------------
                  0x01 - ERROR (error messages)
                  0x02 - INFO (basic messages, low volume)
                  0x04 - VERBOSE (interesting stuff, moderate volume)
                  0x08 - DEBUG (diagnostic, high volume)
```

```
                  0x10 - FUNCS (function entry/exit, very high volume)
                  0x20 - FRAMES (dumps all SMP and GMP frames)
                  0x40 - ROUTING (dump FDB routing information)
                  0x80 - currently unused.
               Without  -vf,  osmtest defaults to ERROR + INFO (0x3)
               Specifying -vf 0 disables all messages Specifying
               -vf 0xFF enables all messages  (see -V) High verbosity
               levels may require increasing the transaction timeout
               with the -t option
-h, --help       Display this usage info then exit.
```

8.3.2 Running osmtest

To run **osmtest** in the default mode, simply enter:

```
host1# osmtest
```

The default mode runs all the flows except for the Quality of Service flow (see Section 8.6).

After installing **opensm** (and if the InfiniBand fabric is stable), it is recommended to run the following command in order to generate the inventory file:

```
host1# osmtest -f c
```

Immediately afterwards, run the following command to test **opensm**:

```
host1# osmtest -f a
```

Finally, it is recommended to occasionally run "**`osmtest -v`**" (with verbosity) to verify that nothing in the fabric has changed.

8.4 Partitions

OpenSM enables the configuration of partitions (PKeys) in an InfiniBand fabric. By default, OpenSM searches for the partitions configuration file under the name `/usr/etc/opensm/partitions.conf`. To change this filename, you can use **opensm** with the '--Pconfig' or '-P' flags.

The default partition is created by OpenSM unconditionally, even when a partition configuration file does not exist or cannot be accessed.

The default partition has a P_Key value of 0x7fff. The port out of which runs OpenSM is assigned full membership in the default partition. All other end-ports are assigned partial membership.

8.4.1 File Format

Notes:

- Line content followed after '#' character is comment and ignored by parser.

General File Format

```
<Partition Definition>:<PortGUIDs list> ;
```

Partition Definition:

```
[PartitionName][=PKey][,flag[=value]][,defmember=full|limited]
```

where

```
PartitionName     string, will be used with logging. When omitted, an           empty
string will be used.
PKey              P_Key value for this partition. Only low 15 bits will
be used. When omitted, P_Key will be autogenerated.
flag              used to indicate IPoIB capability of this partition.
defmember=full|limited
                  specifies default membership for port guid list. Default is limited.
```

Currently recognized flags are:

```
ipoib             indicates that this partition may be used for IPoIB, as a
result IPoIB capable MC group will be created.
rate=<val>        specifies rate for this IPoIB MC group (default is 3 (10GBps))
mtu=<val>         specifies MTU for this IPoIB MC group (default is 4 (2048))
sl=<val>          specifies SL for this IPoIB MC group (default is 0)
scope=<val>       specifies scope for this IPoIB MC group (default is 2 (link local))
```

Note that values for rate, MTU, and scope should be specified as defined in the IBTA specification (for example, mtu=4 for 2048). To use 4K MTU, edit that entry to `"mtu=5"` (5 indicates 4K MTU to that specific partition).

PortGUIDs list:

```
PortGUID          GUID of partition member EndPort. Hexadecimal numbers should start
from 0x, decimal numbers are accepted too.
full or limited   indicates full or limited membership for this port. When omitted (or
unrecognized) limited membership is assumed.
```

There are two useful keywords for PortGUID definition:

- 'ALL' means all end-ports in this subnet
- 'SELF' means subnet manager's port

An empty list means that there are no ports in this partition.

Notes:

- White space is permitted between delimiters ('=', ',',':',';').
- The line can be wrapped after ':' after a Partition Definition and between.
- A PartitionName *does not* need to be unique, but PKey *does* need to be unique.
- If a PKey is repeated then the associated partition configurations will be merged and the first PartitionName will be used (see also next note).
- It is possible to split a partition configuration in more than one definition, but then they PKey should be explicitly specified (otherwise different PKey values will be generated for those definitions).

Examples:

```
Default=0x7fff : ALL, SELF=full ;
NewPartition , ipoib : 0x123456=full, 0x3456789034=limi,  0x2134af2306;

YetAnotherOne = 0x300 : SELF=full ;
YetAnotherOne = 0x300 : ALL=limited ;

ShareIO = 0x80 , defmember=full : 0x123451, 0x123452;
# 0x123453, 0x123454 will be limited
ShareIO = 0x80 : 0x123453, 0x123454, 0x123455=full;
# 0x123456, 0x123457 will be limited
ShareIO   =   0x80   :   defmember=limited   :   0x123456,   0x123457,
0x123458=full;
ShareIO = 0x80 , defmember=full : 0x123459, 0x12345a;
ShareIO  =  0x80  ,  defmember=full  :   0x12345b,   0x12345c=limited,
0x12345d;
```

The following rule is equivalent to how OpenSM used to run prior to the partition manager:

```
Default=0x7fff,ipoib:ALL=full;
```

8.5 Routing Algorithms

OpenSM offers six routing engines:

1. "Min Hop Algorithm"

   Based on the minimum hops to each node where the path length is optimized.

2. "UPDN Algorithm"

   Based on the minimum hops to each node, but it is constrained to ranking rules. This algorithm should be chosen if the subnet is not a pure Fat Tree, and a deadlock may occur due to a loop in the subnet.

3. "Fat-tree Routing Algorithm"

   This algorithm optimizes routing for a congestion-free "shift" communication pattern. It should be chosen if a subnet is a symmetrical Fat Tree of various types, not just a K-ary-N-Tree: non-constant K, not fully staffed, and for any CBB ratio. Similar to UPDN, Fat Tree routing is constrained to ranking rules.

4. "LASH Routing Algorithm"

   Uses InfiniBand virtual layers (SL) to provide deadlock-free shortest-path routing while also distributing the paths between layers. LASH is an alternative deadlock-free, topology-agnostic routing algorithm to the non-minimal UPDN algorithm. It avoids the use of a potentially congested root node.

5. "DOR Routing Algorithm"

   Based on the Min Hop algorithm, but avoids port equalization except for redundant links between the same two switches. This provides deadlock free routes for hypercubes when the fabric is cabled as a hypercube and for meshes when cabled as a mesh.

6. "Torus-2QoS Routing Algorithm"

   Based on the DOR Unicast routing algorithm specialized for 2D/3D torus topologies. Torus-2QoS provides deadlock-free routing while supporting two quality of service (QoS) levels. Additionally,

   it can route around multiple failed fabric links or a single failed fabric switch without introducing deadlocks, and without changing path SLvalues granted before the failure.

   OpenSM provides an optional unicast routing cache (enabled by -A or --ucast_cache options). When enabled, unicast routing cache prevents routing recalculation (which is a heavy task in a large cluster) when there was no topology change detected during the heavy sweep, or when the topology change does not require new routing calculation, e.g. when one or more CAs/RTRs/leaf switches going down, or one or more of these nodes coming back after being down. A very common case that is handled by the unicast routing cache is host reboot, which otherwise would cause two full routing recalculations: one when the host goes down, and the other when the host comes back online.

OpenSM also supports a file method which can load routes from a table – see Modular Routing Engine below.

The basic routing algorithm is comprised of two stages:

1. MinHop matrix calculation. How many hops are required to get from each port to each LID? The algorithm to fill these tables is different if you run standard (min hop) or Up/Down. For standard routing, a "relaxation" algorithm is used to propagate min hop from every destination LID through neighbor switches. For Up/Down routing, a BFS from every target is used.

The BFS tracks link direction (up or down) and avoid steps that will perform up after a down step was used.

2. Once MinHop matrices exist, each switch is visited and for each target LID a decision is made as to what port should be used to get to that LID. This step is common to standard and Up/Down routing. Each port has a counter counting the number of target LIDs going through it. When there are multiple alternative ports with same MinHop to a LID, the one with less previously assigned ports is selected.

   If LMC > 0, more checks are added. Within each group of LIDs assigned to same target port:

   a. Use only ports which have same MinHop
   b. First prefer the ones that go to different systemImageGuid (then the previous LID of the same LMC group)
   c. If none, prefer those which go through another NodeGuid
   d. Fall back to the number of paths method (if all go to same node).

8.5.1 Effect of Topology Changes

If a link is added or removed, OpenSM may not recalculate the routes that do not have to change. A route has to change if the port is no longer UP or no longer the MinHop. When routing changes are performed, the same algorithm for balancing the routes is invoked.

In the case of using the file based routing, any topology changes are currently ignored. The 'file' routing engine just loads the LFTs from the file specified, with no reaction to real topology. Obviously, this will not be able to recheck LIDs (by GUID) for disconnected nodes, and LFTs for non-existent switches will be skipped. Multicast is not affected by 'file' routing engine (this uses min hop tables).

8.5.2 Min Hop Algorithm

The Min Hop algorithm is invoked by default if no routing algorithm is specified. It can also be invoked by specifying '-R minhop'.

The Min Hop algorithm is divided into two stages: computation of min-hop tables on every switch and LFT output port assignment. Link subscription is also equalized with the ability to override based on port GUID. The latter is supplied by:

```
-i <equalize-ignore-guids-file>
-ignore-guids <equalize-ignore-guids-file>
                  This option provides the means to define a set of ports (by guids)
that will be ignored by the link load equalization algorithm.
```

LMC awareness routes based on (remote) system or switch basis.

8.5.3 UPDN Algorithm

The UPDN algorithm is designed to prevent deadlocks from occurring in loops of the subnet. A loop-deadlock is a situation in which it is no longer possible to send data between any two hosts connected through the loop. As such, the UPDN routing algorithm should be send if the subnet is not a pure Fat Tree, and one of its loops may experience a deadlock (due, for example, to high pressure).

The UPDN algorithm is based on the following main stages:

1. Auto-detect root nodes - based on the CA hop length from any switch in the subnet, a statistical histogram is built for each switch (hop num vs number of occurrences). If the histogram reflects a specific column (higher than others) for a certain node, then it is marked as a root node. Since the algorithm is statistical, it may not find any root nodes. The list of the root nodes found by this auto-detect stage is used by the ranking process stage.

The user can override the node list manually

If this stage cannot find any root nodes, and the user did not specify a guid list file, OpenSM defaults back to the Min Hop routing algorithm.

2. Ranking process - All root switch nodes (found in stage 1) are assigned a rank of 0. Using the BFS algorithm, the rest of the switch nodes in the subnet are ranked incrementally. This ranking aids in the process of enforcing rules that ensure loop-free paths.
3. Min Hop Table setting - after ranking is done, a BFS algorithm is run from each (CA or switch) node in the subnet. During the BFS process, the FDB table of each switch node traversed by BFS is updated, in reference to the starting node, based on the ranking rules and guid values.

At the end of the process, the updated FDB tables ensure loop-free paths through the subnet.

Up/Down routing does not allow LID routing communication between switches that are located inside spine “switch systems”. The reason is that there is no way to allow a LID route between them that does not break the Up/Down rule. One ramification of this is that you cannot run SM on switches other than the leaf switches of the fabric.

8.5.3.1 UPDN Algorithm Usage

Activation through OpenSM

- Use '-R updn' option (instead of old '-u') to activate the UPDN algorithm.
- Use '-a <root_guid_file>' for adding an UPDN guid file that contains the root nodes for ranking. If the `-a' option is not used, OpenSM uses its auto-detect root nodes algorithm.

Notes on the guid list file:

- A valid guid file specifies one guid in each line. Lines with an invalid format will be discarded
- The user should specify the root switch guids

8.5.4 Fat-tree Routing Algorithm

The fat-tree algorithm optimizes routing for "shift" communication pattern. It should be chosen if a subnet is a symmetrical or almost symmetrical fat-tree of various types. It supports not just K-ary-N-Trees, by handling for non-constant K, cases where not all leafs (CAs) are present, any Constant Bisectional Ratio (CBB)ratio. As in UPDN, fat-tree also prevents credit-loop-deadlocks.

If the root guid file is not provided ('**-a**' or '**--root_guid_file**' options), the topology has to be pure fat-tree that complies with the following rules:

- Tree rank should be between two and eight (inclusively)
- Switches of the same rank should have the same number of UP-going port groups[1], unless they are root switches, in which case the shouldn't have UP-going ports at all.
- Switches of the same rank should have the same number of DOWN-going port groups, unless they are leaf switches.
- Switches of the same rank should have the same number of ports in each UP-going port group.
- Switches of the same rank should have the same number of ports in each DOWN-going port group.
- All the CAs have to be at the same tree level (rank).

If the root guid file is provided, the topology does not have to be pure fat-tree, and it should only comply with the following rules:

- Tree rank should be between two and eight (inclusively)
- All the Compute Nodes[2] have to be at the same tree level (rank). Note that non-compute node CAs are allowed here to be at different tree ranks.

Topologies that do not comply cause a fallback to min hop routing. Note that this can also occur on link failures which cause the topology to no longer be a “pure” fat-tree.

Note that although fat-tree algorithm supports trees with non-integer CBB ratio, the routing will not be as balanced as in case of integer CBB ratio. In addition to this, although the algorithm allows leaf switches to have any number of CAs, the closer the tree is to be fully populated, the more effective the "shift" communication pattern will be. In general, even if the

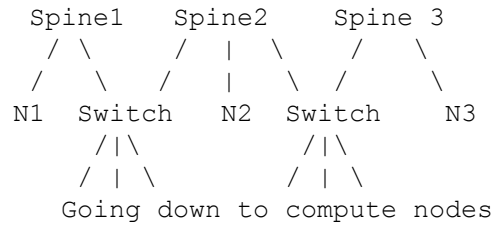
1. Ports that are connected to the same remote switch are referenced as ‘port group’
2. List of compute nodes (CNs) can be specified by ‘-u’ or ‘--cn_guid_file’ OpenSM options.

root list is provided, the closer the topology to a pure and symmetrical fat-tree, the more optimal the routing will be.

The algorithm also dumps compute node ordering file (`opensm-ftree-ca-order.dump`) in the same directory where the OpenSM log resides. This ordering file provides the CN order that may be used to create efficient communication pattern, that will match the routing tables.

8.5.4.1 Routing between non-CN Nodes

The use of the io_guid_file option allows non-CN nodes to be located on different levels in the fat tree. In such case, it is not guaranteed that the Fat Tree algorithm will route between two non-CN nodes. In the scheme below, N1, N2 and N3 are non-CN nodes. Although all the CN have routes to and from them, there will not necessarily be a route between N1,N2 and N3. Such routes would require to use at least one of the switches the wrong way around.



To solve this problem, a list of non-CN nodes can be specified by \'-G\' or \'--io_guid_file\' option. These nodes will be allowed to use switches the wrong way around a specific number of times (specified by \'-H\' or \'--max_reverse_hops\'. With the proper max_reverse_hops and io_guid_file values, you can ensure full connectivity in the Fat Tree. In the scheme above, with a max_reverse_hop of 1, routes will be instanciated between N1<->N2 and N2<->N3. With a max_reverse_hops value of 2, N1,N2 and N3 will all have routes between them.

Using max_reverse_hops creates routes that use the switch in a counter-stream way. This option should never be used to connect nodes with high bandwidth traffic between them! It should only be used to allow connectivity for HA purposes or similar. Also having routes the other way around can cause credit loops.

8.5.4.2 Activation through OpenSM

- Use '-R ftree' option to activate the fat-tree algorithm

LMC > 0 is not supported by fat-tree routing. If this is specified, the default routing algorithm is invoked instead.

8.5.5 LASH Routing Algorithm

LASH is an acronym for LAyered SHortest Path Routing. It is a deterministic shortest path routing algorithm that enables topology agnostic deadlock-free routing within communication networks.

When computing the routing function, LASH analyzes the network topology for the shortest-path routes between all pairs of sources / destinations and groups these paths into virtual layers in such a way as to avoid deadlock.

LASH analyzes routes and ensures deadlock freedom between switch pairs. The link from HCA between and switch does not need virtual layers as deadlock will not arise between switch and HCA.

In more detail, the algorithm works as follows:

1. LASH determines the shortest-path between all pairs of source / destination switches. Note, LASH ensures the same SL is used for all SRC/DST - DST/SRC pairs and there is no guarantee that the return path for a given DST/SRC will be the reverse of the route SRC/DST.
2. LASH then begins an SL assignment process where a route is assigned to a layer (SL) if the addition of that route does not cause deadlock within that layer. This is achieved by maintaining and analysing a channel dependency graph for each layer. Once the potential addition of a path could lead to deadlock, LASH opens a new layer and continues the process.
3. Once this stage has been completed, it is highly likely that the first layers processed will contain more paths than the latter ones. To better balance the use of layers, LASH moves paths from one layer to another so that the number of paths in each layer averages out.

Note that the implementation of LASH in opensm attempts to use as few layers as possible. This number can be less than the number of actual layers available.

In general LASH is a very flexible algorithm. It can, for example, reduce to Dimension Order Routing in certain topologies, it is topology agnostic and fares well in the face of faults.

It has been shown that for both regular and irregular topologies, LASH outperforms Up/Down. The reason for this is that LASH distributes the traffic more evenly through a network, avoiding the bottleneck issues related to a root node and always routes shortest-path.

The algorithm was developed by Simula Research Laboratory.

Use ‘-R lash -Q’ option to activate the LASH algorithm

QoS support has to be turned on in order that SL/VL mappings are used.

LMC > 0 is not supported by the LASH routing. If this is specified, the default routing algorithm is invoked instead.

For open regular cartesian meshes the DOR algorithm is the ideal routing algorithm. For toroidal meshes on the other hand there are routing loops that can cause deadlocks. LASH can be used to route these cases. The performance of LASH can be improved by preconditioning the mesh in cases where there are multiple links connecting switches and also in cases where the switches are not cabled consistently. To invoke this, use '-R lash -Q --do_mesh_analysis'. This will add an additional phase that analyses the mesh to try to determine the dimension and size of a mesh. If it determines that the mesh looks like an open or closed cartesian mesh it reorders the ports in dimension order before the rest of the LASH algorithm runs.

8.5.6 DOR Routing Algorithm

The Dimension Order Routing algorithm is based on the Min Hop algorithm and so uses shortest paths. Instead of spreading traffic out across different paths with the same shortest distance, it chooses among the available shortest paths based on an ordering of dimensions. Each port must be consistently cabled to represent a hypercube dimension or a mesh dimension. Paths are grown from a destination back to a source using the lowest dimension (port) of available paths at each step. This provides the ordering necessary to avoid deadlock. When there are multiple links between any two switches, they still represent only one dimension and traffic is balanced across them unless port equalization is turned off. In the case of hypercubes, the same port must be used throughout the fabric to represent the hypercube dimension and match on both ends of the cable. In the case of meshes, the dimension should consistently use the same pair of ports, one port on one end of the cable, and the other port on the other end, continuing along the mesh dimension.

Use '-R dor' option to activate the DOR algorithm.

8.5.7 Torus-2QoS Routing Algorithm

Torus-2QoS is a routing algorithm designed for large-scale 2D/3D torus fabrics. The torus-2QoS routing engine can provide the following functionality on a 2D/3D torus:

- Free of credit loops routing
- Two levels of QoS, assuming switches support 8 data VLs
- Ability to route around a single failed switch, and/or multiple failed links, without:
  - introducing credit loops
  - changing path SL values
- Very short run times, with good scaling properties as fabric size increases

8.5.7.1 Unicast Routing

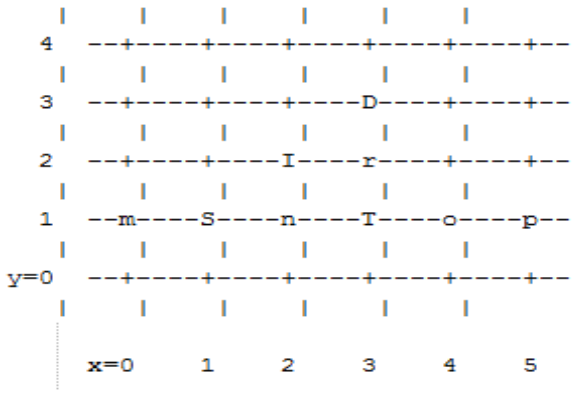
Torus-2 QoS is a DOR-based algorithm that avoids deadlocks that would otherwise occur in a torus using the concept of a dateline for each torus dimension. It encodes into a path SL which datelines the path crosses as follows:

```
sl = 0;
for (d = 0; d < torus_dimensions; d++)
/* path_crosses_dateline(d) returns 0 or 1 */
sl |= path_crosses_dateline(d) << d;
```

For a 3D torus, that leaves one SL bit free, which torus-2 QoS uses to implement two QoS levels. Torus-2 QoS also makes use of the output port dependence of switch SL2VL maps to encode into one VL bit the information encoded in three SL bits. It computes in which torus coordinate direction each inter-switch link "points", and writes SL2VL maps for such ports as follows:

```
for (sl = 0; sl < 16; sl ++)
/* cdir(port) reports which torus coordinate direction a switch port
* "points" in, and returns 0, 1, or 2 */
sl2vl(iport,oport,sl) = 0x1 & (sl >> cdir(oport));
```

Thus, on a pristine 3D torus, i.e., in the absence of failed fabric switches, torus-2 QoS consumes 8 SL values (SL bits 0-2) and 2 VL values (VL bit 0) per QoS level to provide deadlock-free routing on a 3D torus. Torus-2 QoS routes around link failure by "taking the long way around" any 1D ring interrupted by a link failure. For example, consider the 2D 6x5 torus below, where switches are denoted by [+a-zA-Z]:



For a pristine fabric the path from S to D would be S-n-T-r-D. In the event that either link S-n or n-T has failed, torus-2QoS would use the path S-m-p-o-T-r-D.

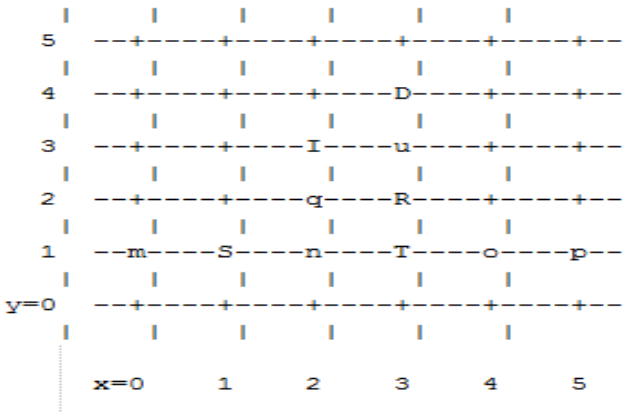
Note that it can do this without changing the path SL value; once the 1D ring m-S-n-T-o-p-m has been broken by failure, path segments using it cannot contribute to deadlock, and the x-direction dateline (between, say, x=5 and x=0) can be ignored for path segments on that ring. One result of this is that torus-2QoS can route around many simultaneous link failures, as long as no 1D ring is broken into disjoint segments. For example, if links n-T and T-o have both failed, that ring has been broken into two disjoint segments, T and o-p-m-S-n. Torus-2QoS checks for such issues, reports if they are found, and refuses to route such fabrics.

Note that in the case where there are multiple parallel links between a pair of switches, torus-2QoS will allocate routes across such links in a round-robin fashion, based on ports at the path destination switch that are active and not used for inter-switch links. Should a link that is one of severalsuch parallel links fail, routes are redistributed across the remaining links. When the last of such a set of parallel links fails, traffic is rerouted as described above.

Handling a failed switch under DOR requires introducing into a path at least one turn that would be otherwise "illegal", i.e. not allowed by DOR rules. Torus-2QoS will introduce such a turn as close as possible to the failed switch in order to route around it. n the above example, suppose switch T has failed, and consider the path from S to D. Torus-2QoS will produce the path S-n-I-r-D, rather than the S-n-T-r-D path for a pristine torus, by introducing an early turn at n. Normal DOR rules will cause traffic arriving at switch I to be forwarded to switch r; for traffic arriving from I due to the "early" turn at n, this will generate an "illegal" turn at I.

Torus-2QoS will also use the input port dependence of SL2VL maps to set VL bit 1 (which would be otherwise unused) for y-x, z-x, and z-y turns, i.e., those turns that are illegal under DOR. This causes the first hop after any such turn to use a separate set of VL values, and prevents deadlock in the presence of a single failed switch. For any given path, only the hops after a turn that is illegal under DOR can contribute to a credit loop that leads to deadlock. So in the example above with failed switch T, the location of the illegal turn at I in the path from S to D requires that any credit loop caused by that turn must encircle the failed switch at T. Thus the second and later hops after the illegal turn at I (i.e., hop r-D) cannot contribute to a credit loop

because they cannot be used to construct a loop encircling T. The hop I-r uses a separate VL, so it cannot contribute to a credit loop encircling T. Extending this argument shows that in addition to being capable of routing around a single switch failure without introducing deadlock, torus-2QoS can also route around multiple failed switches on the condition they are adjacent in the last dimension routed by DOR. For example, consider the following case on a 6x6 2D torus:



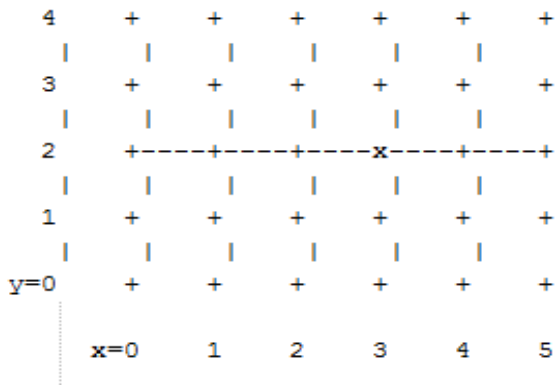
Suppose switches T and R have failed, and consider the path from S to D. Torus-2QoS will generate the path S-n-q-I-u-D, with an illegal turn at switch I, and with hop I-u using a VL with bit 1 set. As a further example, consider a case that torus-2QoS cannot route without deadlock: two failed switches adjacent in a dimension that is not the last dimension routed by DOR; here the failed switches are O and T:

In a pristine fabric, torus-2QoS would generate the path from S to D as S-n-O-T-r-D. With failed switches O and T, torus-2QoS will generate the path S-n-I-q-r-D, with illegal turn at switch I, and with hop I-q using a VL with bit 1 set. In contrast to the earlier examples, the second hop after the illegal turn, q-r, can be used to construct a credit loop encircling the failed switches.

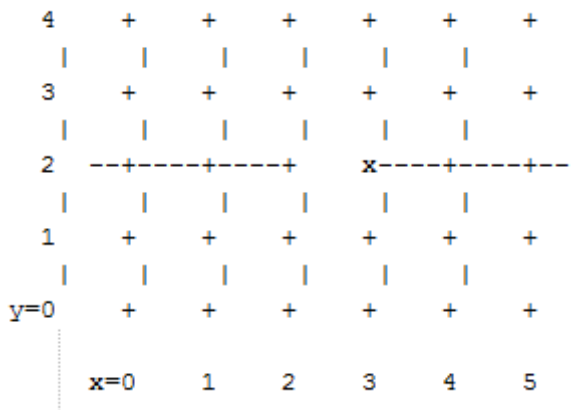
8.5.7.2 Multicast Routing

Since torus-2QoS uses all four available SL bits, and the three data VL bits that are typically available in current switches, there is no way to use SL/VL values to separate multicast traffic from unicast traffic. Thus, torus-2QoS must generate multicast routing such that credit loops can-

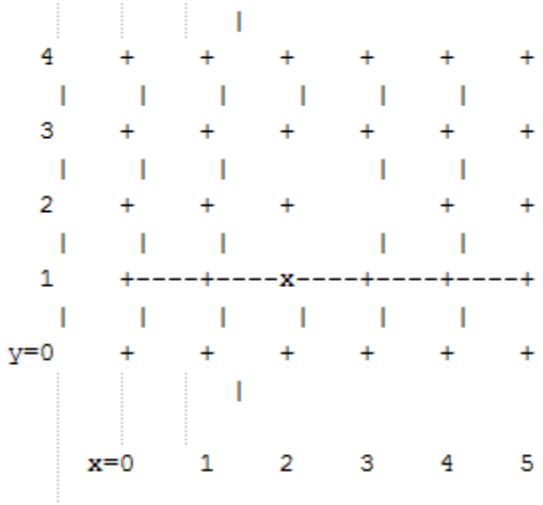
not arise from a combination of multicast and unicast path segments. It turns out that it is possible to construct spanning trees for multicast routing that have that property. For the 2D 6x5 torus example above, here is the full-fabric spanning tree that torus-2QoS will construct, where "x" is the root switch and each "+" is a non-root switch:



For multicast traffic routed from root to tip, every turn in the above spanning tree is a legal DOR turn. For traffic routed from tip to root, and some traffic routed through the root, turns are not legal DOR turns. However, to construct a credit loop, the union of multicast routing on this spanning tree with DOR unicast routing can only provide 3 of the 4 turns needed for the loop. In addition, if none of the above spanning tree branches crosses a dateline used for unicast credit loop avoidance on a torus, and if multicast traffic is confined to SL 0 or SL 8 (recall that torus-2QoS uses SL bit 3 to differentiate QoS level), then multicast traffic also cannot contribute to the "ring" credit loops that are otherwise possible in a torus. Torus-2QoS uses these ideas to create a master spanning tree. Every multicast group spanning tree will be constructed as a subset of the master tree, with the same root as the master tree. Such multicast group spanning trees will in general not be optimal for groups which are a subset of the full fabric. However, this compromise must be made to enable support for two QoS levels on a torus while preventing credit loops. In the presence of link or switch failures that result in a fabric for which torus-2QoS can generate credit-loop-free unicast routes, it is also possible to generate a master spanning tree for multicast that retains the required properties. For example, consider that same 2D 6x5 torus, with the link from (2,2) to (3,2) failed. Torus-2QoS will generate the following master spanning tree:



Two things are notable about this master spanning tree. First, assuming the x dateline was between x=5 and x=0, this spanning tree has a branch that crosses the dateline. However, just as for unicast, crossing a dateline on a 1D ring (here, the ring for y=2) that is broken by a failure cannot contribute to a torus credit loop. Second, this spanning tree is no longer optimal even for multicast groups that encompass the entire fabric. That, unfortunately, is a compromise that must be made to retain the other desirable properties of torus-2QoS routing. In the event that a single switch fails, torus-2QoS will generate a master spanning tree that has no "extra" turns by appropriately selecting a root switch. In the 2D 6x5 torus example, assume now that the switch at (3,2), i.e. the root for a pristine fabric, fails. Torus-2QoS will generate the following master spanning tree for that case:



Assuming the y dateline was between y=4 and y=0, this spanning tree has a branch that crosses a dateline. However, again this cannot contribute to credit loops as it occurs on a 1D ring (the ring for x=3) that is broken by a failure, as in the above example.

8.5.7.3 Torus Topology Discovery

The algorithm used by torus-2QoS to construct the torus topology from the undirected graph representing the fabric requires that the radix of each dimension be configured via torus-2QoS.conf. It also requires that the torus topology be "seeded"; for a 3D torus this requires configuring four switches that define the three coordinate directions of the torus. Given this starting information, the algorithm is to examine the cube formed by the eight switch locations bounded by the corners (x,y,z) and (x+1,y+1,z+1). Based on switches already placed into the torus topology at some of these locations, the algorithm examines 4-loops of interswitch links to find the one that is consistent with a face of the cube of switch locations, and adds its swiches to the discovered topology in the correct locations.

Because the algorithm is based on examining the topology of 4-loops of links, a torus with one or more radix-4 dimensions requires extra initial seed configuration. See torus-2QoS.conf(5) for details. Torus-2QoS will detect and report when it has insufficient configuration for a torus with radix-4 dimensions.

In the event the torus is significantly degraded, i.e., there are many missing switches or links, it may happen that torus-2QoS is unable to place into the torus some switches and/or links that were discovered in the fabric, and will generate a warning in that case. A similar condition

occurs if torus-2QoS is misconfigured, i.e., the radix of a torus dimension as configured does not match the radix of that torus dimension as wired, and many switches/links in the fabric will not be placed into the torus.

8.5.7.4 Quality Of Service Configuration

OpenSM will not program switchs and channel adapters with SL2VL maps or VL arbitration configuration unless it is invoked with -Q. Since torus-2QoS depends on such functionality for correct operation, always invoke OpenSM with -Q when torus-2QoS is in the list of routing engines. Any quality of service configuration method supported by OpenSM will work with torus-2QoS, subject to the following limitations and considerations. For all routing engines supported by OpenSM except torus-2QoS, there is a one-to-one correspondence between QoS level and SL. Torus-2QoS can only support two quality of service levels, so only the high-order bit of any SL value used for unicast QoS configuration will be honored by torus-2QoS. For multicast QoS configuration, only SL values 0 and 8 should be used with torus-2QoS.

Since SL to VL map configuration must be under the complete control of torus-2QoS, any configuration via qos_sl2vl, qos_swe_sl2vl, etc., must and will be ignored, and a warning will be generated. Torus-2QoS uses VL values 0-3 to implement one of its supported QoS levels, and VL values 4-7 to implement the other. Hard-to-diagnose application issues may arise if traffic is not delivered fairly across each of these two VL ranges. Torus-2QoS will detect and warn if VL arbitration is configured unfairly across VLs in the range 0-3, and also in the range 4-7. Note that the default OpenSM VL arbitration configuration does not meet this constraint, so all torus-2QoS users should configure VL arbitration via qos_vlarb_high, qos_vlarb_low, etc.

8.5.7.5 Operational Considerations

Any routing algorithm for a torus IB fabric must employ path SL values to avoid credit loops. As a result, all applications run over such fabrics must perform a path record query to obtain the correct path SL for connection setup. Applications that use rdma_cm for connection setup will automatically meet this requirement.

If a change in fabric topology causes changes in path SL values required to route without credit loops, in general all applications would need to repath to avoid message deadlock. Since torus-2QoS has the ability to reroute after a single switch failure without changing path SL values, repathing by running applications is not required when the fabric is routed with torus-2QoS.

Torus-2QoS can provide unchanging path SL values in the presence of subnet manager failover provided that all OpenSM instances have the same idea of dateline location. See torus-2QoS.conf(5) for details. Torus-2QoS will detect configurations of failed switches and links that prevent routing that is free of credit loops, and will log warnings and refuse to route. If "no_fallback" was configured in the list of OpenSM routing engines, then no other routing engine will attempt to route the fabric. In that case all paths that do not transit the failed components will continue to work, and the subset of paths that are still operational will continue to remain free of credit loops. OpenSM will continue to attempt to route the fabric after every sweep interval, and after any change (such as a link up) in the fabric topology. When the fabric components are repaired, full functionality will be restored. In the event OpenSM was configured to allow some other engine to route the fabric if torus-2QoS fails, then credit loops and message deadlock are likely if torus-2QoS had previously routed the fabric successfully. Even if the other engine is capable of routing a torus without credit loops, applications that built connections with path SL values granted under torus-2QoS will likely experience message deadlock under routing generated by a different engine, unless they repath. To verify that a torus fabric is routed free of credit loops, use ibdmchk to analyze data collected via ibdiagnet -vlr.

8.5.7.6 Torus-2QoS Configuration File Syntax

The file torus-2QoS.conf contains configuration information that is specific to the OpenSM routing engine torus-2QoS. Blank lines and lines where the first non-whitespace character is "#" are ignored. A token is any contiguous group of non-whitespace characters. Any tokens on a line following the recognized configuration tokens described below are ignored.

```
[torus|mesh] x_radix[m|M|t|T] y_radix[m|M|t|T] z_radix[m|M|t|T]
```

Either torus or mesh must be the first keyword in the configuration, and sets the topology that torus-2QoS will try to construct. A 2D topology can be configured by specifying one of x_radix, y_radix, or z_radix as 1. An individual dimension can be configured as mesh (open) or torus (looped) by suffixing its radix specification with one of m, M, t, or T. Thus, "mesh 3T 4 5" and "torus 3 4M 5M" both specify the same topology.

Note that although torus-2QoS can route mesh fabrics, its ability to route around failed components is severely compromised on such fabrics. A failed fabric componentis very likely to cause a disjoint ring; see UNICAST ROUTING in torus-2QoS(8).

```
xp_link sw0_GUID sw1_GUID
yp_link sw0_GUID sw1_GUID
zp_link sw0_GUID sw1_GUID
xm_link sw0_GUID sw1_GUID
ym_link sw0_GUID sw1_GUID
zm_link sw0_GUID sw1_GUID
```

These keywords are used to seed the torus/mesh topology. For example, "xp_link 0x2000 0x2001" specifies that a link from the switch with node GUID 0x2000 to the switch with node GUID 0x2001 would point in the positive x direction, while "xm_link 0x2000 0x2001" specifies that a link from the switch with node GUID 0x2000 to the switch with node GUID 0x2001 would point in the negative x direction. All the link keywords for a given seed must specify the same "from" switch.

In general, it is not necessary to configure both the positive and negative directions for a given coordinate; either is sufficient. However, the algorithm used for topology discovery needs extra information for torus dimensions of radix four (see TOPOLOGY DISCOVERY in torus-2QoS(8)). For such cases both the positive and negative coordinate directions must be specified.

Based on the topology specified via the torus/mesh keyword, torus-2QoS will detect and log when it has insufficient seed configuration.

```
x_dateline position
y_dateline position
z_dateline position
```

In order for torus-2QoS to provide the guarantee that path SL values do not change under any conditions for which it can still route the fabric, its idea of dateline position must not change relative to physical switch locations. The dateline keywords provide the means to configure such behavior.

The dateline for a torus dimension is always between the switch with coordinate 0 and the switch with coordinate radix-1 for that dimension. By default, the common switch in a torus seed is taken as the origin of the coordinate system used to describe switch location. The position param-

eter for a dateline keyword moves the origin (and hence the dateline) the specified amount relative to the common switch in a torus seed.

```
next_seed
```

If any of the switches used to specify a seed were to fail torus-2QoS would be unable to complete topology discovery successfully. The next_seed keyword specifies that the following link and dateline keywords apply to a new seed specification.

For maximum resiliency, no seed specification should share a switch with any other seed specification. Multiple seed specifications should use dateline configuration to ensure that torus-2QoS can grant path SL values that are constant, regardless of which seed was used to initiate topology discovery.

portgroup_max_ports max_ports - This keyword specifies the maximum number of parallel inter-switch links, and also the maximum number of host ports per switch, that torus-2QoS can accommodate. The default value is 16. Torus-2QoS will log an error message during topology discovery if this parameter needs to be increased. If this keyword appears multiple times, the last instance prevails.

port_order p1 p2 p3 ... - This keyword specifies the order in which CA ports on a destination switch are visited when computing routes. When the fabric contains switches connected with multiple parallel links, routes are distributed in a round-robin fashion across such links, and so changing the order that CA ports are visited changes the distribution of routes across such links. This may be advantageous for some specific traffic patterns.

The default is to visit CA ports in increasing port order on destination switches. Duplicate values in the list will be ignored.

EXAMPLE

```
# Look for a 2D (since x radix is one) 4x5 torus.
torus 1 4 5
# y is radix-4 torus dimension, need both
# ym_link and yp_link configuration.
yp_link 0x200000 0x200005 # sw @ y=0,z=0 -> sw @ y=1,z=0
ym_link 0x200000 0x20000f # sw @ y=0,z=0 -> sw @ y=3,z=0
# z is not radix-4 torus dimension, only need one of
# zm_link or zp_link configuration.
zp_link 0x200000 0x200001 # sw @ y=0,z=0 -> sw @ y=0,z=1
next_seed
yp_link 0x20000b 0x200010 # sw @ y=2,z=1 -> sw @ y=3,z=1
ym_link 0x20000b 0x200006 # sw @ y=2,z=1 -> sw @ y=1,z=1
zp_link 0x20000b 0x20000c # sw @ y=2,z=1 -> sw @ y=2,z=2
y_dateline -2 # Move the dateline for this seed
z_dateline -1 # back to its original position.
# If OpenSM failover is configured, for maximum resiliency
# one instance should run on a host attached to a switch
# from the first seed, and another instance should run
# on a host attached to a switch from the second seed.
# Both instances should use this torus-2QoS.conf to ensure
# path SL values do not change in the event of SM failover.
# port_order defines the order on which the ports would be
# chosen for routing.
port_order 7 10 8 11 9 12 25 28 26 29 27 30
```

8.5.8 Routing Chains

As of today, as it is impossible to configure each part of the fabric to be routed using different routing engines, a fabric can be routed using only one routing engine at a time. The routing chains feature is offering a solution that enables one to configure different parts of the fabric and define a different routing engine to route each of them. The routings are done in a sequence (hence the name "chains") and any node in the fabric that is configured in more than one part is left with the routing updated by the last routing engine it was a part of.

8.5.8.1 Configuring Routing Chains

The configuration for the routing chains feature consists of the following steps:

1. Define the port groups.
2. Define topologies based on previously defined port groups.
3. Define configuration files for each routing engine.
4. Define routing engine chains over previously defined topologies and configuration files.

8.5.8.1.1 Defining Port Groups

The basic idea behind the port groups is the ability to divide the fabric into sub-groups and give each group an identifier that can be used to relate to all nodes in this group. The port groups is a separate feature from the routing chains, but is a mandatory prerequisite for it. In addition, it is used to define the participants in each of the routing algorithms.

Defining a Port Group Policy File

In order to define a port group policy file, set the parameter 'pgrp_policy_file' in the opensm configuration file.

```
pgrp_policy_file /etc/opensm/conf/port_groups_policy_file
```

Configuring a Port Group Policy

The port groups policy file details the port groups in the fabric. The policy file should be composed of one or more paragraphs that define a group. Each paragraph should begin with the line 'port-group' and end with the line 'end-port-group'.

For example:

```
port-group
…port group qualifiers…
end-port-group
```

Port Group Qualifiers

Unlike the port group's beginning and ending which do not require a colon, all qualifiers must end with a colon (':'). Also - a colon is a predefined mark that must not be used inside qualifier values. An inclusion of a colon in the name or the use of a port group, will result in the policy's failure.

Table 19 - Port Group Qualifiers

Parameter	Description	Example
name	Each group must have a name. Without a name qualifier, the policy fails.	name: grp1
use	'use' is an optional qualifier that one can define in order to describe the usage of this port group (if undefined, an empty string is used as a default).	use: first port group

Rule Qualifier

There are several qualifiers used to describe a rule that determines which ports will be added to the group. Each port group policy must contain exactly one rule qualifier (If no rules exist, no ports can be chosen. More than one rule, on the other hand, will cause a conflict).

Table 20 - Port Group Qualifiers

Parameter	Description	Example
guid list	Comma separated list of guids to include in the group. If no specific physical ports were configured, all physical ports of the guid are chosen. However, for each guid, one can detail specific physical ports to be included in the group. This can be done using the following syntax: • Specify a specific port in a guid to be chosen port-guid: 0x283@3 • Specify a specific list of ports in a guid to be chosen port-guid: 0x286@1/5/7 • Specify a specific range of ports in a guid to be chosen port-guid: 0x289@2-5 • Specify a list of specific ports and ports ranges in a guid to be chosen port-guid: 0x289@2-5/7/9-13/18 • Complex rule port-guid: 0x283@5-8/12/14, 0x286, 0x289/6/8/12	port-guid: 0x283, 0x286, 0x289
port guid range	It is possible to configure a range of guids to be chosen to the group. However, while using the range qualifier, it is impossible to detail specific physical ports. **Note:** A list of ranges cannot be specified. The below example is invalid and will cause the policy to fail: port-guid-range: 0x283-0x289, 0x290-0x295	port-guid-range: 0x283-0x289

Table 20 - Port Group Qualifiers

Parameter	Description	Example
port name	One can configure a list of hostnames as a rule. Hosts with a node description that is built out of these hostnames will be chosen. Since the node description contains the network card index as well, one might also specify a network card index and a physical port to be chosen. For example, the given configuration will cause only physical port 2 of a host with the node description 'kuku HCA-1' to be chosen. `port` and `hca_idx` parameters are optional. If the port is unspecified, all physical ports are chosen. If `hca_idx` is unspecified, all card numbers are chosen. Specifying a hostname is mandatory. One can configure a list of `hostname/port/hca_idx` sets in the same qualifier as follows: port-name: hostname=kuku; port=2; hca_idx=1 , hostname=host1; port=3, hostname=host2 **Note:** `port-name` qualifier is not relevant for switches, but for HCA's only.	port-name: hostname=kuku; port=2; hca_idx=1
port regexp	One can define a regular expression so that only nodes with a matching node description will be chosen to the group	port-regexp: SW*
	It is possible to specify one physical port to be chosen for matching nodes (there is no option to define a list or a range of ports). The given example will cause only nodes that match physical port 3 to be added to the group.	port-regexp: SW*:3
union rule	It is possible to define a rule that unites two different port groups. This means that all ports from both groups will be included in the united group.	union-rule: grp1, grp2
subtract rule	One can define a rule that subtracts one port group from another. The given rule, for example, will cause all the ports which are a part of grp1, but not included in grp2, to be chosen. In subtraction (unlike union), the order does matter, since the purpose is to subtract the second group from the first one. There is no option to define more than two groups for union/subtraction. However, one can unite/subtract groups which are a union or a subtraction themselves, as shown in the port groups policy file example.	subtract-rule: grp1, grp2

Predefined Port Groups

There are 3 predefined, automatically created port groups that are available for use, yet cannot be defined in the policy file (if a group in the policy is configured with the name of one of these predefined groups, the policy fails) -

- ALL - a group that includes all nodes in the fabric
- ALL_SWITCHES - a group that includes all switches in the fabric.
- ALL_CAS - a group that includes all HCA's in the fabric.

Port Groups Policy Examples

```
port-group
name: grp3
use: Subtract of groups grp1 and grp2
subtract-rule:  grp1, grp2
end-port-group

port-group
name: grp1
port-guid: 0x281, 0x282, 0x283
end-port-group

port-group
name: grp2
port-guid-range: 0x282-0x286
end-port-group

port-group
name: grp4
port-name: hostname=kika port=1 hca_idx=1
end-port-group

port-group
name: grp3
union-rule: grp3, grp4
end-port-group
```

8.5.8.2 Defining a Topologies Policy File

In order to define a port group policy file, set the parameter 'topo_policy_file' in the opensm configuration file.

```
topo_policy_file /etc/opensm/conf/topo_policy_file.cfg
```

Configuring a Topology Policy

The topologies policy file details a list of topologies. The policy file should be composed of one or more paragraphs which define a topology. Each paragraph should begin with the line 'topology' and end with the line 'end-topology'.

For example:

```
topology
…topology qualifiers…
end-topology
```

Topology Qualifiers

Unlike topology and end-topology which do not require a colon, all qualifiers must end with a colon (':'). Also - a colon is a predefined mark that must not be used inside qualifier values. An inclusion of a column in the qualifier values will result in the policy's failure.

All topology qualifiers are mandatory. Absence of any of the below qualifiers will cause the policy parsing to fail.

Table 21 - Topology Qualifiers

Parameter	Description	Example
id	Topology ID. Legal Values – any positive value. Must be unique.	id: 1
sw-grp	Name of the port group that includes all switches and switch ports to be used in this topology.	sw-grp: ys_switches
hca-grp	Name of the port group that includes all HCA's to be used in this topology.	hca-grp: ys_hosts

Configuration File per Routing Engine

Each engine in the routing chain can be provided by its own configuration file. Routing engine configuration file is the fraction of parameters defined in the main opensm configuration file.

Some rules should be applied when defining a particular configuration file for a routing engine:

- Parameters that are not specified in specific routing engine configuration file are inherited from the main opensm configuration file.
- The following configuration parameters are taking effect only in the main opensm configuration file:
  - qos and qos_* settings like (vl_arb, sl2vl, etc.)
  - lmc
  - routing_engine

8.5.8.3 Defining a Routing Chain Policy File

In order to define a port group policy file, set the parameter 'rch_policy_file' in the opensm configuration file.

```
rch_policy_file /etc/opensm/conf/chains_policy_file
```

First Routing Engine in the Chain

The first unicast engine in a routing chain must include all switches and HCA's in the fabric (topology id must be 0). The path-bit parameter value is path-bit 0 and it cannot be changed.

Configuring a Routing Chains Policy

The routing chains policy file details the routing engines (and their fallback engines) used for the fabric's routing. The policy file should be composed of one or more paragraphs which defines an engine (or a fallback engine). Each paragraph should begin with the line 'unicast-step' and end with the line 'end-unicast-step'.

For example:

```
unicast-step
…routing engine qualifiers…
end-unicast-step
```

Routing Engine Qualifiers

Unlike unicast-step and end-unicast-step which do not require a colon, all qualifiers must end with a colon (':'). Also - a colon is a predefined mark that must not be used inside qualifier values. An inclusion of a colon in the qualifier values will result in the policy's failure.

Table 22 - Routing Engine Qualifiers

Parameter	Description	Example
id	'id' is man**dato**ry. Without an id qualifier for each engine, the policy fails. • Legal values – size_t value (0 is illegal). • The engines in the policy chain are set according to an ascending id order, so it is highly crucial to verify that the id that is given to the engines match the order in which you would like the engines to be set.	is: 1
engine	This is a mandatory qualifier that describes the routing algorithm used within this unicast step. Currently, on the first phase of routing chains, legal values are minhop/ftree/updn.	engine: minhop
use	This is an optional qualifier that enables one to describe the usage of this unicast step. If undefined, an empty string is used as a default.	use: ftree routing for for yellow stone nodes
config	This is an optional qualifier that enables one to define a separate opensm config file for a specific unicast step. If undefined, all parameters are taken from main opensm configuration file.	config: /etc/config/opensm2.cfg

Table 22 - Routing Engine Qualifiers

Parameter	Description	Example
topology	Define the topology that this engine uses. • Legal value – id of an existing topology that is defined in topologies policy (or zero that represents the entire fabric and not a specific topology). • Default value – If unspecified, a routing engine will relate to the entire fabric (as if topology zero was defined). • Notice: The first routing engine (the engine with the lowest id) MUST be configured with topology: 0 (entire fabric) or else, the routing chain parser will fail.	topology: 1
fallback-to	This is an optional qualifier that enables one to define the current unicast step as a fallback to another unicast step. This can be done by defining the id of the unicast step that this step is a fallback to. • If undefined, the current unicast step is not a fallback. • If the value of this qualifier is a non-existent engine id, this step will be ignored. • A fallback step is meaningless if the step it is a fallback to did not fail. • It is impossible to define a fallback to a fallback step (such definition will be ignored)	-
path-bit	This is an optional qualifier that enables one to define a specific lid offset to be used by the current unicast step. Setting lmc > 0 in main opensm configuration file is a prerequisite for assigning specific path-bit for the routing engine. Default value is 0 (if path-bit is not specified)	Path-bit: 1

Dump Files per Routing Engine

Each routing engine on the chain will dump its own data files if the appropriate log_flags is set (for instance 0x43).

The files that are dumped by each engine are:

- opensm-lid-matrix.dump
- opensm-lfts.dump
- opensm.fdbs
- opensm-subnet.lst

These files should contain the relevant data for each engine topology.

sl2vl and mcfdbs files are dumped only once for the entire fabric and NOT by every routing engine.

- Each engine concatenates its ID and routing algorithm name in its dump files names, as follows:
  - opensm-lid-matrix.2.minhop.dump
  - opensm.fdbs.3.ftree
  - opensm-subnet.4.updn.lst
- In case that a fallback routing engine is used, both the routing engine that failed and the fallback engine that replaces it, dump their data.

  If, for example, engine 2 runs ftree and it has a fallback engine with 3 as its id that runs minhop, one should expect to find 2 sets of dump files, one for each engine:
  - opensm-lid-matrix.2.ftree.dump
  - opensm-lid-matrix.3.minhop.dump
  - opensm.fdbs.2.ftree
  - opensm.fdbs.3.munhop

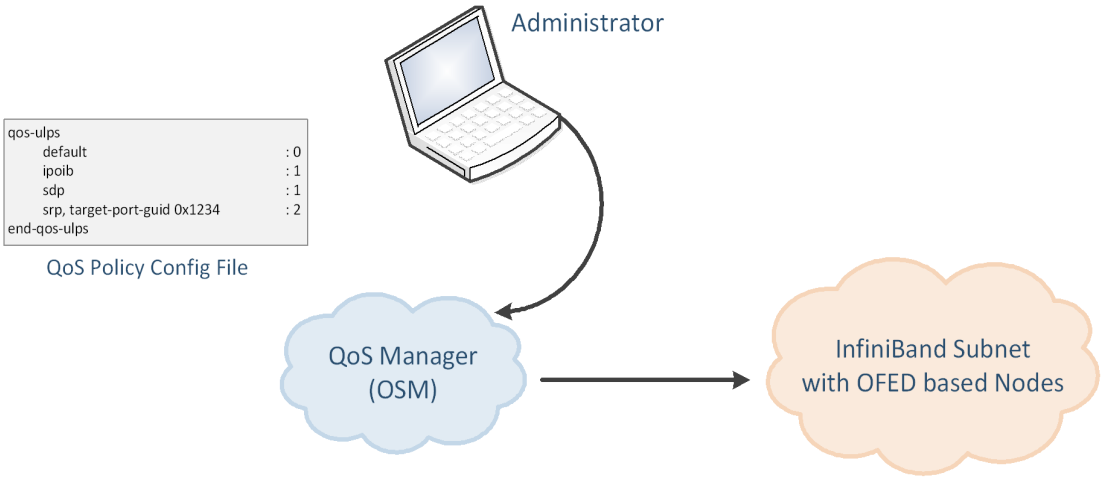
8.6 Quality of Service Management in OpenSM

8.6.1 Overview

When Quality of Service (QoS) in OpenSM is enabled (using the '-Q' or '--qos' flags), OpenSM looks for a QoS Policy file. During fabric initialization and at every heavy sweep, OpenSM parses the QoS policy file, applies its settings to the discovered fabric elements, and enforces the provided policy on client requests. The overall flow for such requests is as follows:

- The request is matched against the defined matching rules such that the QoS Level definition is found
- Given the QoS Level, a path(s) search is performed with the given restrictions imposed by that level

Figure 6: QoS Manager



There are two ways to define QoS policy:

- Advanced – the advanced policy file syntax provides the administrator various ways to match a PathRecord/MultiPathRecord (PR/MPR) request, and to enforce various QoS constraints on the requested PR/MPR
- Simple – the simple policy file syntax enables the administrator to match PR/MPR requests by various ULPs and applications running on top of these ULPs

8.6.2 Advanced QoS Policy File

The QoS policy file has the following sections:

I) Port Groups (denoted by port-groups)

This section defines zero or more port groups that can be referred later by matching rules (see below). Port group lists ports by:

- Port GUID
- Port name, which is a combination of NodeDescription and IB port number
- PKey, which means that all the ports in the subnet that belong to partition with a given PKey belong to this port group
- Partition name, which means that all the ports in the subnet that belong to partition with a given name belong to this port group
- Node type, where possible node types are: CA, SWITCH, ROUTER, ALL, and SELF (SM's port).

II) QoS Setup (denoted by qos-setup)

This section describes how to set up SL2VL and VL Arbitration tables on various nodes in the fabric. However, this is not supported in OFED. SL2VL and VLArb tables should be configured in the OpenSM options file (default location - /var/cache/opensm/opensm.opts).

III) QoS Levels (denoted by qos-levels)

Each QoS Level defines Service Level (SL) and a few optional fields:

- MTU limit

- Rate limit
- PKey
- Packet lifetime

When path(s) search is performed, it is done with regards to restriction that these QoS Level parameters impose. One QoS level that is mandatory to define is a DEFAULT QoS level. It is applied to a PR/MPR query that does not match any existing match rule. Similar to any other QoS Level, it can also be explicitly referred by any match rule.

IV) QoS Matching Rules (denoted by qos-match-rules)

Each PathRecord/MultiPathRecord query that OpenSM receives is matched against the set of matching rules. Rules are scanned in order of appearance in the QoS policy file such as the first match takes precedence.

Each rule has a name of QoS level that will be applied to the matching query. A default QoS level is applied to a query that did not match any rule.

Queries can be matched by:

- Source port group (whether a source port is a member of a specified group)
- Destination port group (same as above, only for destination port)
- PKey
- QoS class
- Service ID

To match a certain matching rule, PR/MPR query has to match ALL the rule's criteria. However, not all the fields of the PR/MPR query have to appear in the matching rule.

For instance, if the rule has a single criterion - Service ID, it will match any query that has this Service ID, disregarding rest of the query fields. However, if a certain query has only Service ID (which means that this is the only bit in the PR/MPR component mask that is on), it will not match any rule that has other matching criteria besides Service ID.

8.6.3 Simple QoS Policy Definition

Simple QoS policy definition comprises of a single section denoted by qos-ulps. Similar to the advanced QoS policy, it has a list of match rules and their QoS Level, but in this case a match rule has only one criterion - its goal is to match a certain ULP (or a certain application on top of this ULP) PR/MPR request, and QoS Level has only one constraint - Service Level (SL).

The simple policy section may appear in the policy file in combine with the advanced policy, or as a stand-alone policy definition. See more details and list of match rule criteria below.

8.6.4 Policy File Syntax Guidelines

- Leading and trailing blanks, as well as empty lines, are ignored, so the indentation in the example is just for better readability.
- Comments are started with the pound sign (#) and terminated by EOL.
- Any keyword should be the first non-blank in the line, unless it's a comment.
- Keywords that denote section/subsection start have matching closing keywords.

- Having a QoS Level named "DEFAULT" is a must - it is applied to PR/MPR requests that didn't match any of the matching rules.
- Any section/subsection of the policy file is optional.

8.6.5 Examples of Advanced Policy File

As mentioned earlier, any section of the policy file is optional, and the only mandatory part of the policy file is a default QoS Level.

Here's an example of the shortest policy file:

```
qos-levels
    qos-level
        name: DEFAULT
        sl: 0
    end-qos-level
end-qos-levels
```

Port groups section is missing because there are no match rules, which means that port groups are not referred anywhere, and there is no need defining them. And since this policy file doesn't have any matching rules, PR/MPR query will not match any rule, and OpenSM will enforce default QoS level. Essentially, the above example is equivalent to not having a QoS policy file at all.

The following example shows all the possible options and keywords in the policy file and their syntax:

```
#
# See the comments in the following example.
# They explain different keywords and their meaning.
#
port-groups

    port-group # using port GUIDs
        name: Storage
        # "use" is just a description that is used for logging
        #  Other than that, it is just a comment
        use: SRP Targets
        port-guid: 0x10000000000001, 0x10000000000005-0x1000000000FFFA
        port-guid: 0x1000000000FFFF
    end-port-group

    port-group
        name: Virtual Servers
        # The syntax of the port name is as follows:
        #   "node_description/Pnum".
        # node_description is compared to the NodeDescription of the node,
        # and "Pnum" is a port number on that node.
        port-name: vs1 HCA-1/P1, vs2 HCA-1/P1
```

```
        end-port-group

        # using partitions defined in the partition policy
        port-group
            name: Partitions
            partition: Part1
            pkey: 0x1234
        end-port-group

        # using node types: CA, ROUTER, SWITCH, SELF (for node that runs SM)
        # or ALL (for all the nodes in the subnet)
        port-group
            name: CAs and SM
            node-type: CA, SELF
        end-port-group

    end-port-groups

    qos-setup
        # This section of the policy file describes how to set up SL2VL and VL
        # Arbitration tables on various nodes in the fabric.
        # However, this is not supported in OFED - the section is parsed
        # and ignored. SL2VL and VLArb tables should be configured in the
        # OpenSM options file (by default - /var/cache/opensm/opensm.opts).
    end-qos-setup

    qos-levels

        # Having a QoS Level named "DEFAULT" is a must - it is applied to
        # PR/MPR requests that didn't match any of the matching rules.
        qos-level
            name: DEFAULT
            use: default QoS Level
            sl: 0
        end-qos-level

        # the whole set: SL, MTU-Limit, Rate-Limit, PKey, Packet Lifetime
        qos-level
            name: WholeSet
            sl: 1
            mtu-limit: 4
            rate-limit: 5
            pkey: 0x1234
            packet-life: 8
        end-qos-level
```

```
end-qos-levels

# Match rules are scanned in order of their apperance in the policy file.
# First matched rule takes precedence.
qos-match-rules

    # matching by single criteria: QoS class
    qos-match-rule
        use: by QoS class
        qos-class: 7-9,11
        # Name of qos-level to apply to the matching PR/MPR
        qos-level-name: WholeSet
    end-qos-match-rule

    # show matching by destination group and service id
    qos-match-rule
        use: Storage targets
        destination: Storage
        service-id: 0x10000000000001, 0x10000000000008-0x10000000000FFF
        qos-level-name: WholeSet
    end-qos-match-rule

    qos-match-rule
        source: Storage
        use: match by source group only
        qos-level-name: DEFAULT
    end-qos-match-rule

    qos-match-rule
        use: match by all parameters
        qos-class: 7-9,11
        source: Virtual Servers
        destination: Storage
        service-id: 0x0000000000010000-0x000000000001FFFF
        pkey: 0x0F00-0x0FFF
        qos-level-name: WholeSet
    end-qos-match-rule

end-qos-match-rules
```

8.6.6 Simple QoS Policy - Details and Examples

Simple QoS policy match rules are tailored for matching ULPs (or some application on top of a ULP) PR/MPR requests. This section has a list of per-ULP (or per-application) match rules and the SL that should be enforced on the matched PR/MPR query.

Match rules include:

- Default match rule that is applied to PR/MPR query that didn't match any of the other match rules
- SDP
- SDP application with a specific target TCP/IP port range
- SRP with a specific target IB port GUID
- RDS
- IPoIB with a default PKey
- IPoIB with a specific PKey
- Any ULP/application with a specific Service ID in the PR/MPR query
- Any ULP/application with a specific PKey in the PR/MPR query
- Any ULP/application with a specific target IB port GUID in the PR/MPR query

Since any section of the policy file is optional, as long as basic rules of the file are kept (such as no referring to nonexisting port group, having default QoS Level, etc), the simple policy section (qos-ulps) can serve as a complete QoS policy file.

The shortest policy file in this case would be as follows:

```
qos-ulps
    default  : 0 #default SL
end-qos-ulps
```

It is equivalent to the previous example of the shortest policy file, and it is also equivalent to not having policy file at all. Below is an example of simple QoS policy with all the possible keywords:

```
qos-ulps
default               :0 # default SL
sdp, port-num 30000   :0 # SL for application running on
                           # top of SDP when a destination
                           # TCP/IPport is 30000
sdp, port-num 10000-20000     : 0
sdp                   :1 # default SL for any other
                           # application running on top of SDP
rds                   :2 # SL for RDS traffic
ipoib, pkey 0x0001    :0 # SL for IPoIB on partition with
                           # pkey 0x0001
ipoib                 :4 # default IPoIB partition,
                           # pkey=0x7FFF
any, service-id 0x6234:6 # match any PR/MPR query with a
                           # specific Service ID
any, pkey 0x0ABC      :6 # match any PR/MPR query with a
                           # specific PKey
srp, target-port-guid 0x1234  : 5 # SRP when SRP Target is located
                           # on a specified IB port GUID
any, target-port-guid 0x0ABC-0xFFFFF : 6 # match any PR/MPR query
                           # with a specific target port GUID
```

```
end-qos-ulps
```

Similar to the advanced policy definition, matching of PR/MPR queries is done in order of appearance in the QoS policy file such as the first match takes precedence, except for the "default" rule, which is applied only if the query didn't match any other rule. All other sections of the QoS policy file take precedence over the qos-ulps section. That is, if a policy file has both qos-match-rules and qos-ulps sections, then any query is matched first against the rules in the qos-match-rules section, and only if there was no match, the query is matched against the rules in qos-ulps section.

Note that some of these match rules may overlap, so in order to use the simple QoS definition effectively, it is important to understand how each of the ULPs is matched.

8.6.6.1 IPoIB

IPoIB query is matched by PKey or by destination GID, in which case this is the GID of the multicast group that OpenSM creates for each IPoIB partition.

Default PKey for IPoIB partition is 0x7fff, so the following three match rules are equivalent:

```
ipoib      :<SL>
ipoib, pkey 0x7fff : <SL>
any,   pkey 0x7fff : <SL>
```

8.6.6.2 SDP

SDP PR query is matched by Service ID. The Service-ID for SDP is 0x000000000001PPPP, where PPPP are 4 hex digits holding the remote TCP/IP Port Number to connect to. The following two match rules are equivalent:

```
sdp                                              :<SL>
any, service-id 0x0000000000010000-0x000000000001ffff : <SL>
```

8.6.6.3 RDS

Similar to SDP, RDS PR query is matched by Service ID. The Service ID for RDS is 0x000000000106PPPP, where PPPP are 4 hex digits holding the remote TCP/IP Port Number to connect to. Default port number for RDS is 0x48CA, which makes a default Service-ID 0x00000000010648CA. The following two match rules are equivalent:

```
rds                          :<SL>
any, service-id 0x00000000010648CA : <SL>
```

8.6.6.4 SRP

Service ID for SRP varies from storage vendor to vendor, thus SRP query is matched by the target IB port GUID. The following two match rules are equivalent:

```
srp, target-port-guid 0x1234  : <SL>
any, target-port-guid 0x1234  : <SL>
```

Note that any of the above ULPs might contain target port GUID in the PR query, so in order for these queries not to be recognized by the QoS manager as SRP, the SRP match rule (or any match rule that refers to the target port guid only) should be placed at the end of the qos-ulps match rules.

8.6.6.5 MPI

SL for MPI is manually configured by MPI admin. OpenSM is not forcing any SL on the MPI traffic, and that's why it is the only ULP that did not appear in the qos-ulps section.

8.6.7 SL2VL Mapping and VL Arbitration

OpenSM cached options file has a set of QoS related configuration parameters, that are used to configure SL2VL mapping and VL arbitration on IB ports. These parameters are:

- Max VLs: the maximum number of VLs that will be on the subnet
- High limit: the limit of High Priority component of VL Arbitration table (IBA 7.6.9)
- VLArb low table: Low priority VL Arbitration table (IBA 7.6.9) template
- VLArb high table: High priority VL Arbitration table (IBA 7.6.9) template
- SL2VL: SL2VL Mapping table (IBA 7.6.6) template. It is a list of VLs corresponding to SLs 0-15 (Note that VL15 used here means drop this SL).

There are separate QoS configuration parameters sets for various target types: CAs, routers, switch external ports, and switch's enhanced port 0. The names of such parameters are prefixed by "qos_<type>_" string. Here is a full list of the currently supported sets:

- qos_ca_ - QoS configuration parameters set for CAs.
- qos_rtr_ - parameters set for routers.
- qos_sw0_ - parameters set for switches' port 0.
- qos_swe_ - parameters set for switches' external ports.

Here's the example of typical default values for CAs and switches' external ports (hard-coded in OpenSM initialization):

```
qos_ca_max_vls 15
qos_ca_high_limit 0
qos_ca_vlarb_high 0:4,1:0,2:0,3:0,4:0,5:0,6:0,7:0,8:0,9:0,10:0,11:0,12:0,13:0,14:0
qos_ca_vlarb_low 0:0,1:4,2:4,3:4,4:4,5:4,6:4,7:4,8:4,9:4,10:4,11:4,12:4,13:4,14:4
qos_ca_sl2vl 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,7
qos_swe_max_vls 15
qos_swe_high_limit 0
qos_swe_vlarb_high 0:4,1:0,2:0,3:0,4:0,5:0,6:0,7:0,8:0,9:0,10:0,11:0,12:0,13:0,14:0
qos_swe_vlarb_low 0:0,1:4,2:4,3:4,4:4,5:4,6:4,7:4,8:4,9:4,10:4,11:4,12:4,13:4,14:4
qos_swe_sl2vl 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,7
```

VL arbitration tables (both high and low) are lists of VL/Weight pairs. Each list entry contains a VL number (values from 0-14), and a weighting value (values 0-255), indicating the number of 64 byte units (credits) which may be transmitted from that VL when its turn in the arbitration occurs. A weight of 0 indicates that this entry should be skipped. If a list entry is programmed for VL15 or for a VL that is not supported or is not currently configured by the port, the port may either skip that entry or send from any supported VL for that entry.

Note, that the same VLs may be listed multiple times in the High or Low priority arbitration tables, and, further, it can be listed in both tables. The limit of high-priority VLArb table (qos_<type>_high_limit) indicates the number of high-priority packets that can be transmitted without an opportunity to send a low-priority packet. Specifically, the number of bytes that can be sent is high_limit times 4K bytes.

A high_limit value of 255 indicates that the byte limit is unbounded.

If the 255 value is used, the low priority VLs may be starved.

A value of 0 indicates that only a single packet from the high-priority table may be sent before an opportunity is given to the low-priority table.

Keep in mind that ports usually transmit packets of size equal to MTU. For instance, for 4KB MTU a single packet will require 64 credits, so in order to achieve effective VL arbitration for packets of 4KB MTU, the weighting values for each VL should be multiples of 64.

Below is an example of SL2VL and VL Arbitration configuration on subnet:

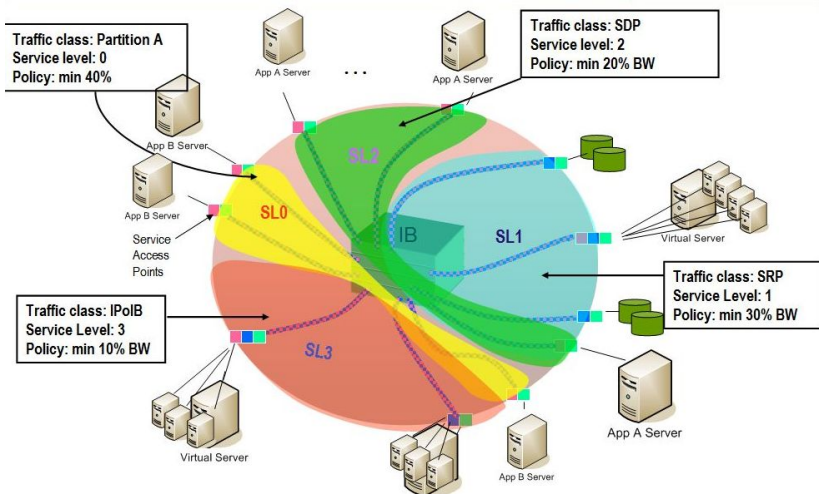
```
qos_ca_max_vls 15
qos_ca_high_limit 6
qos_ca_vlarb_high 0:4
qos_ca_vlarb_low 0:0,1:64,2:128,3:192,4:0,5:64,6:64,7:64
qos_ca_sl2vl 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,7
qos_swe_max_vls 15
qos_swe_high_limit 6
qos_swe_vlarb_high 0:4
qos_swe_vlarb_low 0:0,1:64,2:128,3:192,4:0,5:64,6:64,7:64
qos_swe_sl2vl 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,7
```

In this example, there are 8 VLs configured on subnet: VL0 to VL7. VL0 is defined as a high priority VL, and it is limited to 6 x 4KB = 24KB in a single transmission burst. Such configuration would suilt VL that needs low latency and uses small MTU when transmitting packets. Rest of VLs are defined as low priority VLs with different weights, while VL4 is effectively turned off.

8.6.8 Deployment Example

Figure 7 shows an example of an InfiniBand subnet that has been configured by a QoS manager to provide different service levels for various ULPs.

Figure 7: Example QoS Deployment on InfiniBand Subnet



8.7 QoS Configuration Examples

The following are examples of QoS configuration for different cluster deployments. Each example provides the QoS level assignment and their administration via OpenSM configuration files.

8.7.1 Typical HPC Example: MPI and Lustre

Assignment of QoS Levels

- MPI
  - Separate from I/O load
  - Min BW of 70%
- Storage Control (Lustre MDS)
  - Low latency
- Storage Data (Lustre OST)
  - Min BW 30%

Administration

- MPI is assigned an SL via the command line

```
host1# mpirun -sl 0
```

- OpenSM QoS policy file

In the following policy file example, replace OST* and MDS* with the real port GUIDs.

```
qos-ulps
```

```
    default                                                    :0 # default SL (for
MPI)
    any, target-port-guid OST1,OST2,OST3,OST4:1 # SL for Lustre OST
    any, target-port-guid MDS1,MDS2                            :2 # SL for Lustre
MDS
end-qos-ulps
```

- OpenSM options file

```
 qos_max_vls 8
 qos_high_limit 0
 qos_vlarb_high 2:1
 qos_vlarb_low 0:96,1:224
 qos_sl2vl 0,1,2,3,4,5,6,7,15,15,15,15,15,15,15,15
```

8.7.2 EDC SOA (2-tier): IPoIB and SRP

The following is an example of QoS configuration for a typical enterprise data center (EDC) with service oriented architecture (SOA), with IPoIB carrying all application traffic and SRP used for storage.

QoS Levels

- Application traffic
  - IPoIB (UD and CM) and SDP
  - Isolated from storage
  - Min BW of 50%
- SRP
  - Min BW 50%
  - Bottleneck at storage nodes

Administration

- OpenSM QoS policy file

In the following policy file example, replace SRPT* with the real SRP Target port GUIDs.

```
qos-ulps
    default                                :0
    ipoib                                  :1
    sdp                                    :1
    srp, target-port-guid SRPT1,SRPT2,SRPT3 :2
end-qos-ulps
```

- OpenSM options file

```
qos_max_vls 8
 qos_high_limit 0
 qos_vlarb_high 1:32,2:32
 qos_vlarb_low 0:1,
 qos_sl2vl 0,1,2,3,4,5,6,7,15,15,15,15,15,15,15,15
```

8.7.3 EDC (3-tier): IPoIB, RDS, SRP

The following is an example of QoS configuration for an enterprise data center (EDC), with IPoIB carrying all application traffic, RDS for database traffic, and SRP used for storage.

QoS Levels

- Management traffic (ssh)
  - IPoIB management VLAN (partition A)
  - Min BW 10%
- Application traffic
  - IPoIB application VLAN (partition B)
  - Isolated from storage and database
  - Min BW of 30%
- Database Cluster traffic
  - RDS
  - Min BW of 30%
- SRP
  - Min BW 30%
  - Bottleneck at storage nodes

Administration

- OpenSM QoS policy file

In the following policy file example, replace SRPT* with the real SRP Initiator port GUIDs.

```
qos-ulps
    default                          :0
    ipoib, pkey 0x8001               :1
    ipoib, pkey 0x8002               :2
    rds                              :3
    srp, target-port-guid SRPT1, SRPT2, SRPT3  : 4
end-qos-ulps
```

- OpenSM options file

```
qos_max_vls 8
qos_high_limit 0
qos_vlarb_high 1:32,2:96,3:96,4:96
qos_vlarb_low 0:1
qos_sl2vl 0,1,2,3,4,5,6,7,15,15,15,15,15,15,15,15
```

- Partition configuration file

```
Default=0x7fff,     ipoib : ALL=full;
PartA=0x8001, sl=1, ipoib : ALL=full;
```

8.8 Adaptive Routing

8.8.1 Overview

Adaptive Routing is at beta stage.

Adaptive Routing (AR) enables the switch to select the output port based on the port's load. AR supports two routing modes:

- Free AR: No constraints on output port selection.
- Bounded AR: The switch does not change the output port during the same transmission burst. This mode minimizes the appearance of out-of-order packets.

Adaptive Routing Manager enables and configures Adaptive Routing mechanism on fabric switches. It scans all the fabric switches, deduces which switches support Adaptive Routing and configures the AR functionality on these switches.

Currently, Adaptive Routing Manager supports only link aggregation algorithm. Adaptive Routing Manager configures AR mechanism to allow switches to select output port out of all the ports that are linked to the same remote switch. This algorithm suits any topology with several links between switches. Especially, it suits 3D torus/mesh, where there are several link in each direction of the X/Y/Z axis.

If some switches do not support AR, they will slow down the AR Manager as it may get timeouts on the AR-related queries to these switches.

8.8.2 Installing the Adaptive Routing

Adaptive Routing Manager is a Subnet Manager plug-in, i.e. it is a shared library (libarmgr.so) that is dynamically loaded by the Subnet Manager. Adaptive Routing Manager is installed as a part of Mellanox OFED installation.

8.8.3 Running Subnet Manager with Adaptive Routing Manager

Adaptive Routing (AR) Manager can be enabled/disabled through SM options file.

8.8.3.1 Enabling Adaptive Routing

To enable Adaptive Routing, perform the following:

1. Create the Subnet Manager options file. Run:

```
opensm -c <options-file-name>
```

2. Add '**armgr**' to the '**event_plugin_name**' option in the file:

```
# Event plugin name(s)
event_plugin_name armgr
```

3. Run Subnet Manager with the new options file:

```
opensm -F <options-file-name>
```

Adaptive Routig Manager can read options file with various configuration parameters to fine-tune AR mechanism and AR Manager behavior. Default location of the AR Manager options file is **/etc/opensm/ar_mgr.conf**.

To provide an alternative location, please perform the following:

1. Add '**armgr --conf_file <ar-mgr-options-file-name>**' to the '**event_plugin_options**' option in the file **# Options** string that would be passed to the plugin(s) **event_plugin_options armgr --conf_file <ar-mgr-options-file-name>**
2. Run Subnet Manager with the new options file:

```
opensm -F <options-file-name>
```

See an example of AR Manager options file with all the default values in "Example of Adaptive Routing Manager Options File" on page 215.

8.8.3.2 Disabling Adaptive Routing

There are two ways to disable Adaptive Routing Manager:

1. By disabling it explicitly in the Adaptive Routing configuration file.
2. By removing the 'armgr' option from the Subnet Manager options file.

Adaptive Routing mechanism is automatically disabled once the switch receives setting of the usual linear routing table (LFT).

Therefore, no action is required to clear Adaptive Routing configuration on the switches if you do not wish to use Adaptive Routing.

8.8.4 Querying Adaptive Routing Tables

When Adaptive Routing is active, the content of the usual Linear Forwarding Routing Table on the switch is invalid, thus the standard tools that query LFT (e.g. smpquery, dump_lfts.sh, and others) cannot be used. To query the switch for the content of its Adaptive Routing table, use the 'smparquery' tool that is installed as a part of the Adaptive Routing Manager package. To see its usage details, run 'smparquery -h'.

8.8.5 Adaptive Routing Manager Options File

The default location of the AR Manager options file is /etc/opensm/ar_mgr.conf. To set an alternative location, please perform the following:

1. Add 'armgr --conf_file **<ar-mgr-options-file-name>** to the **event_plugin_option**' option in the file **# Options** string that would be passed to the plugin(s) **event_plugin_options armgr --conf_file <ar-mgr-options-file-name>**
2. Run Subnet Manager with the new options file:

```
opensm -F <options-file-name>'
```

AR Manager options file contains two types of parameters:

1. General options - Options which describe the AR Manager behavior and the AR parameters that will be applied to all the switches in the fabric.
2. Per-switch options - Options which describe specific switch behavior.

Note the following:

- Adaptive Routing configuration file is case sensitive.
- You can specify options for nonexisting switch GUID. These options will be ignored until a switch with a matching GUID will be added to the fabric.
- Adaptive Routing configuration file is parsed every AR Manager cycle, which in turn is executed at every heavy sweep of the Subnet Manager.
- If the AR Manager fails to parse the options file, default settings for all the options will be used.

8.8.5.1 General AR Manager Options

Table 23 - Adaptive Routing Manager Options File

Option File	Description	Values
ENABLE: <true|false>	Enable/disable Adaptive Routing on fabric switches. Note that if a switch was identified by AR Manager as device that does not support AR, AR Manager will not try to enable AR on this switch. If the firmware of this switch was updated to support the AR, the AR Manager will need to be restarted (by restarting Subnet Manager) to allow it to configure the AR on this switch. This option can be changed on-the-fly.	Default: true
AR_MODE: <bounded|free>	Adaptive Routing Mode: • free: no constraints on output port selection • bounded: the switch does not change the output port during the same transmission burst. This mode minimizes the appearance of out-of-order packets This option can be changed on-the-fly.	Default: bounded
AGEING_TIME: <usec>	Applicable to bounded AR mode only. Specifies how much time there should be no traffic in order for the switch to declare a transmission burst as finished and allow changing the output port for the next transmission burst (32-bit value). This option can be changed on-the-fly.	Default: 30
MAX_ERRORS: <N> ERROR_WINDOW : <N>	When number of errors exceeds 'MAX_ERRORS' of send/receive errors or timeouts in less than 'ERROR_WINDOW' seconds, the AR Manager will abort, returning control back to the Subnet Manager. This option can be changed on-the-fly.	Values for both options: [0-0xffff] • MAX_ERRORS = 0: zero tollerance - abort configuration on first error. Default:10 • ERROR_WINDOW = 0: mechanism disabled - no error checking. Default: 5
LOG_FILE: <full path>	AR Manager log file. This option can be changed on-the-fly.	Default: /var/log/armgr.log
LOG_SIZE: <size in MB>	This option defines maximal AR Manager log file size in MB. The logfile will be truncated and restarted upon reaching this limit. This option cannot be changed on-the-fly.	0: unlimited log file size. Default: 5

8.8.5.1.1 Per-switch AR Options

A user can provide per-switch configuration options with the following syntax:

```
SWITCH <GUID> {
                  <switch option 1>;
                  <switch option 2>;
                  ...
                  }
```

The following are the per-switch options:

Table 24 - Adaptive Routing Manager Pre-Switch Options File

Option File	Description	Values
ENABLE: <true|false>	Allows you to enable/disable the AR on this switch. If the general ENABLE option value is set to 'false', then this per-switch option is ignored. This option can be changed on the fly.	Default: true
AGEING_TIME: <usec>	Applicable to bounded AR mode only. Specifies how much time there should be no traffic in order for the switch to declare a transmission burst as finished and allow changing the output port for the next transmission burst (32-bit value). In the pre-switch options file this option refers to the particular switch only This option can be changed on-the-fly.	Default: 30

8.8.5.1.2 Example of Adaptive Routing Manager Options File

```
ENABLE: true;
 LOG_FILE: /tmp/ar_mgr.log;
 LOG_SIZE: 100;
 MAX_ERRORS: 10;
 ERROR_WINDOW: 5;

SWITCH 0x12345 {
ENABLE: true;
AGEING_TIME: 77;
 }

SWITCH 0x0002c902004050f8 {
AGEING_TIME: 44;
 }

SWITCH 0xabcde {
ENABLE: false;
 }
```

8.9 Congestion Control

8.9.1 Congestion Control Overview

Congestion Control Manager is a Subnet Manager (SM) plug-in, i.e. it is a shared library (libcc-mgr.so) that is dynamically loaded by the Subnet Manager. Congestion Control Manager is installed as part of Mellanox OFED installation.

The Congestion Control mechanism controls traffic entry into a network and attempts to avoid oversubscription of any of the processing or link capabilities of the intermediate nodes and networks. Additionally, is takes resource reducing steps by reducing the rate of sending packets. Congestion Control Manager enables and configures Congestion Control mechanism on fabric nodes (HCAs and switches).

8.9.2 Running OpenSM with Congestion Control Manager

Congestion Control (CC) Manager can be enabled/disabled through SM options file. To do so, perform the following:

1. Create the file. Run:

```
opensm -c <options-file-name>'
```

2. Find the '**event_plugin_name**' option in the file, and add '**ccmgr**' to it.

```
# Event plugin name(s)
event_plugin_name ccmgr
```

3. Run the SM with the new options file: '**opensm -F <options-file-name>**'

Once the Congestion Control is enabled on the fabric nodes, to completely disable Congestion Control, you will need to actively turn it off. Running the SM w/o the CC Manager is not sufficient, as the hardware still continues to function in accordance to the previous CC configuration.

For further information on how to turn OFF CC, please refer to Section 8.9.3, “Configuring Congestion Control Manager”, on page 216

8.9.3 Configuring Congestion Control Manager

Congestion Control (CC) Manager comes with a predefined set of setting. However, you can fine-tune the CC mechanism and CC Manager behavior by modifying some of the options. To do so, perform the following:

1. Find the '**event_plugin_options**' option in the SM options file, and add the following:

```
conf_file <cc-mgr-options-file-name>':
# Options string that would be passed to the plugin(s)
event_plugin_options ccmgr --conf_file <cc-mgr-options-file-name>
```

2. Run the SM with the new options file: '**opensm -F <options-file-name>**'

To turn CC OFF, set 'enable' to 'FALSE' in the Congestion Control Manager configuration file, and run OpenSM ones with this configuration.

For the full list of CC Manager options with all the default values, See "Configuring Congestion Control Manager" on page 216.

For further details on the list of CC Manager options, please refer to the IB spec.

8.9.4 Configuring Congestion Control Manager Main Settings

To fine-tune CC mechanism and CC Manager behavior, and set the CC manager main settings, perform the following:

- To enables/disables Congestion Control mechanism on the fabric nodes, set the following parameter:

```
enable
```

  - The values are: **<TRUE | FALSE>**.
  - The default is: **True**

- CC manager configures CC mechanism behavior based on the fabric size. The larger the fabric is, the more aggressive CC mechanism is in its response to congestion. To manually modify CC manager behavior by providing it with an arbitrary fabric size, set the following parameter:

```
num_hosts
```

  - The values are: **[0-48K]**.
  - The default is: **0** (base on the CCT calculation on the current subnet size)

- The smaller the number value of the parameter, the faster HCAs will respond to the congestion and will throttle the traffic. Note that if the number is too low, it will result in suboptimal bandwidth. To change the mean number of packets between marking eligible packets with a FECN, set the following parameter:

```
marking_rate
```

  - The values are: **[0-0xffff]**.
  - The default is: **0xa**

- You can set the minimal packet size that can be marked with FECN. Any packet less than this size [bytes] will not be marked with FECN. To do so, set the following parameter:

```
packet_size
```

  - The values are: **[0-0x3fc0]**.
  - The default is: **0x200**

- When number of errors exceeds 'max_errors' of send/receive errors or timeouts in less than 'error_window' seconds, the CC MGR will abort and will allow OpenSM to proceed. To do so, set the following parameter:

```
max_errors
error_window
```

  - The values are:

    max_errors = 0: zero tollerance - abort configuration on first error
    error_window = 0: mechanism disabled - no error checking.[0-48K]

  - The default is: `5`

8.9.4.1 Congestion Control Manager Options File

Table 25 - Congestion Control Manager General Options File

Option File	Description	Values
enable	Enables/disables Congestion Control mechanism on the fabric nodes.	Values: <TRUE | FALSE> Default: True
num_hosts	Indicates the number of nodes. The CC table values are calculated based on this number.	Values: [0-48K] Default: 0 (base on the CCT calculation on the current subnet size)

Table 26 - Congestion Control Manager Switch Options File

Option File	Description	Values
threshold	Indicates how aggressive the congestion marking should be.	[0-0xf] • 0 - no packet marking, • 0xf - very aggressive Default: 0xf
marking_rate	The mean number of packets between marking eligible packets with a FECN	Values: [0-0xffff] Default: 0xa
packet_size	Any packet less than this size [bytes] will not be marked with FECN.	Values: [0-0x3fc0] Default: 0x200

Table 27 - Congestion Control Manager CA Options File

Option File	Desctiption	Values
port_control	Specifies the Congestion Control attribute for this port	Values: • 0 - QP based congestion control, • 1 - SL/Port based congestion control Default: 0

Table 27 - Congestion Control Manager CA Options File

Option File	Desctiption	Values
ca_control_map	An array of sixteen bits, one for each SL. Each bit indicates whether or not the corresponding SL entry is to be modified.	Values: 0xffff
ccti_increase	Sets the CC Table Index (CCTI) increase.	Default: 1
trigger_threshold	Sets the trigger threshold.	Default: 2
ccti_min	Sets the CC Table Index (CCTI) minimum.	Default: 0
cct	Sets all the CC table entries to a specified value . The first entry will remain 0, whereas last value will be set to the rest of the table.	Values: <comma-separated list> Default: 0 When the value is set to 0, the CCT calculation is based on the number of nodes.
ccti_timer	Sets for all SL's the given ccti timer.	Default: 0 When the value is set to 0, the CCT calculation is based on the number of nodes.

Table 28 - Congestion Control Manager CC MGR Options File

Option File	Desctiption	Values
max_errors error_window	When number of errors exceeds 'max_errors' of send/receive errors or timeouts in less than 'error_window' seconds, the CC MGR will abort and will allow OpenSM to proceed.	Values: • max_errors = 0: zero tollerance - abort configuration on first error. • error_window = 0: mechanism disabled - no error checking. Default: 5
cc_statistics_cycle	Enables CC MGR to collect statistics from all nodes every cc_statistics_cycle [seconds]	Default: 0 When the value is set to 0, no statistics are collected.

9 InfiniBand Fabric Utilities

This section first describes common configuration, interface, and addressing for all the tools in the package. Then it provides detailed descriptions of the tools themselves including: operation, synopsis and options descriptions, error codes, and examples.

9.1 Common Configuration, Interface and Addressing

Topology File (Optional)

An InfiniBand fabric is composed of switches and channel adapter (HCA/TCA) devices. To identify devices in a fabric (or even in one switch system), each device is given a GUID (a MAC equivalent). Since a GUID is a non-user-friendly string of characters, it is better to alias it to a meaningful, user-given name. For this objective, the IB Diagnostic Tools can be provided with a "topology file", which is an optional configuration file specifying the IB fabric topology in user-given names.

For diagnostic tools to fully support the topology file, the user may need to provide the local system name (if the local hostname is not used in the topology file).

To specify a topology file to a diagnostic tool use one of the following two options:

1. On the command line, specify the file name using the option '`-t <topology file name>`'
2. Define the environment variable IBDIAG_TOPO_FILE

To specify the local system name to an diagnostic tool use one of the following two options:

1. On the command line, specify the system name using the option '`-s <local system name>`'
2. Define the environment variable IBDIAG_SYS_NAME

9.2 InfiniBand Interface Definition

The diagnostic tools installed on a machine connect to the IB fabric by means of an HCA port through which they send MADs. To specify this port to an IB diagnostic tool use one of the following options:

1. On the command line, specify the port number using the option '-p <local port number>' (see below)
2. Define the environment variable IBDIAG_PORT_NUM

In case more than one HCA device is installed on the local machine, it is necessary to specify the device's index to the tool as well. For this use on of the following options:

1. On the command line, specify the index of the local device using the following option: '-i <index of local device>'
2. Define the environment variable IBDIAG_DEV_IDX

9.3 Addressing

This section applies to the ibdiagpath tool only. A tool command may require defining the destination device or port to which it applies.

The following addressing modes can be used to define the IB ports:

- Using a Directed Route to the destination: (Tool option '-d')

  This option defines a directed route of output port numbers from the local port to the destination.

- Using port LIDs: (Tool option '-l'):

  In this mode, the source and destination ports are defined by means of their LIDs. If the fabric is configured to allow multiple LIDs per port, then using any of them is valid for defining a port.

- Using port names defined in the topology file: (Tool option '-n')

  This option refers to the source and destination ports by the names defined in the topology file. (Therefore, this option is relevant only if a topology file is specified to the tool.) In this mode, the tool uses the names to extract the port LIDs from the matched topology, then the tool operates as in the '-l' option.

9.4 Diagnostic Utilities

The diagnostic utilities described in this chapter provide means for debugging the connectivity and status of InfiniBand (IB) devices in a fabric.

9.4.1 ibdiagnet (of ibutils2) - IB Net Diagnostic

This version of ibdiagnet is included in the ibutils2 package, and it is run by default after installing Mellanox OFED. To use this ibdiagnet version, run: ibdiagnet

Please see ibutils2_release_notes.txt for additional information and known issues.

ibdiagnet scans the fabric using directed route packets and extracts all the available information regarding its connectivity and devices. It then produces the following files in the output directory (which is defined by the -o option described below).

Synopsis

```
[-i|--device <dev-name>] [-p|--port <port-num>] [-g|--guid <GUID in hex>] [--vlr <file>]
[-r|--routing] [-u|--fat_tree] [-o|--output_path <directory>] [--skip <stage>]
[--skip_plugin <library name>] [--pc] [-P|--counter <<PM>=<value>>]
[--pm_pause_time <seconds>] [--ber_test] [--ber_use_data] [--ber_thresh <value>]
[--extended_speeds <dev-type>] [--pm_per_lane] [--ls <2.5|5|10|14|25|FDR10>]
[--lw <1x|4x|8x|12x>] [-w|--write_topo_file <file name>] [-t|--topo_file <file>]
[--out_ibnl_dir <directory>]   [--screen_num_errs <num>] [--smp_window <num>]
[--gmp_window <num>] [--max_hops <max-hops>] [-V|--version] [-h|--help] [-H|--deep_help]
```

Options

-i|--device <dev-name>	Specifies the name of the device of the port used to connect to the IB fabric (in case of multiple devices on he local system).
-p|--port <port-num>	Specifies the local device's port number used to connect to the IB fabric.
-g|--guid <GUID in hex>	Specifies the local port GUID value of the port used to connect to the IB fabric. If GUID given is 0 than ibdiagnet displays a list of possible port GUIDs and waits for user input.
--vlr <file>	Specifies opensm-path-records.dump file path, src-dst to SL mapping generated by SM plugin. ibdiagnet will use this mapping for MADs sending and credit loop check (if -r option selected).
-r|--routing	Provides a report of the fabric qualities.
-u|--fat_tree	Indicates that UpDown credit loop checking should be done against automatically determined roots.
-o|--output_path <directory>	Specifies the directory where the output files will be placed.(default="/var/tmp/ibdiagnet2/").
--skip <stage>	Skip the executions of the given stage. Applicable skip stages:(all | dup_guids | dup_node_desc | lids | links | sm | pm | nodes_info | speed_width_check | pkey | aguid).
--skip_plugin <library name>	Skip the load of the given library name. Applicable skip plugins:(libibdiagnet_cable_diag_plugin | libibdiagnet_cable_diag_plugin-2.1.1).
--pc	Reset all the fabric PM counters.
-P|--counter <<PM>=<value>>	If any of the provided PM is greater then its provided value than print it.
--pm_pause_time <seconds>	Specifies the seconds to wait between first counters sample and second counters sample. If seconds given is 0 than no second counters sample will be done. (default=1).
--ber_test	Provides a BER test for each port. Calculate BER for each port and check no BER value has exceeds the BER threshold. (default threshold="10^-12").
--ber_use_data	Indicates that BER test will use the received data for calculation.
--ber_thresh <value>	Specifies the threshold value for the BER test. The reciprocal number of the BER should be provided. Example: for 10^-12 than value need to be 1000000000000 or 0xe8d4a51000 (10^12).If threshold given is 0 than all BER values for all ports will be reported.

```
--extended_speeds <dev-type>    Collect and test port extended speeds counters. dev-type: (sw
                                | all).
--pm_per_lane                   List all counters per lane (when available).
--ls <2.5|5|10|14|25|FDR10>     Specifies the expected link speed.
--lw <1x|4x|8x|12x>             Specifies the expected link width.
-w|--write_topo_file <file      Write out a topology file for the discovered topology.
name>
-t|--topo_file <file>           Specifies the topology file name.
--out_ibnl_dir <directory>      The topology file custom system definitions(ibnl) directory.
--screen_num_errs <num>         Specifies the threshold for printing errors to screen.
                                (default=5).
--smp_window <num>              Max smp MADs on wire. (default=8).
--gmp_window <num>              Max gmp MADs on wire. (default=128).
--max_hops <max-hops>           Specifies the maximum hops for the discovery pro-
                                cess.(default=64).
-V|--version                    Prints the version of the tool.
-h|--help                       Prints help information (without plugins help if exists).
-H|--deep_help                  Prints deep help information (including plugins help).
```

Output Files

Table 29 lists the ibdiagnet output files that are placed under /var/tmp/ibdiagnet2.

Table 29 - ibdiagnet (of ibutils2) Output Files

Output File	Description
ibdiagnet2.lst	Fabric links in LST format
ibdiagnet2.sm	Subnet Manager
ibdiagnet2.pm	Ports Counters
ibdiagnet2.fdbs	Unicast FDBs
ibdiagnet2.mcfdbs	Multicast FDBx
ibdiagnet2.nodes_info	Information on nodes
ibdiagnet2.db_csv	ibdiagnet internal database

ibdiagnet2.pm file example:

```
-------------------------------------------------
Port=1 Lid=0x0017 GUID=0x0002c9000000004d Device=43132 Port Name=S0002c9000000004e/
N0002c9000000004d/P1
-------------------------------------------------
link_down_counter=0x00000000
link_error_recovery_counter=0x00000000
symbol_error_counter=0x00000000
port_rcv_remote_physical_errors=0x00000000
port_rcv_errors=0x00000000
```

```
port_xmit_discard=0x00000000
port_rcv_switch_relay_errors=0x00000000
excessive_buffer_errors=0x00000000
local_link_integrity_errors=0x00000000
port_rcv_constraint_errors=0x00000000
port_xmit_constraint_errors=0x00000000
vl15_dropped=0x00000000
port_xmit_data=0x00000000
port_rcv_data=0x00000000
port_xmit_pkts=0x00000000
port_rcv_pkts=0x00000000
port_xmit_wait=0x00000000
port_xmit_data_extended=0x0000000000000000
port_rcv_data_extended=0x0000000000000000
port_xmit_pkts_extended=0x0000000000000000
port_rcv_pkts_extended=0x0000000000000000
port_unicast_xmit_pkts=0x0000000000000000
port_unicast_rcv_pkts=0x0000000000000000
port_multicast_xmit_pkts=0x0000000000000000
port_multicast_rcv_pkts=0x0000000000000000
retransmission_per_sec=0x0000000000000000
```

Example to report in the log:

```
Port Counters
-I- Ports counters retrieving finished successfully
-I- Going to sleep for 1 seconds until next counters sample
-I- Ports counters retrieving (second time) finished successfully
-E- Ports counters value Check finished with errors
-E- lid=0x0001 dev=23108 H-11/U1/U1/P1
        Performance Monitor counter         : Value
        retransmission_per_sec              : 600        (threshold=500)
…
```

An ibdiagnet run performs the following stages:

- Fabric discovery
- Duplicated GUIDs detection
- Links in INIT state and unresponsive links detection
- Counters fetch
- Error counters check
- Routing checks
- Link width and speed checks
- Alias GUIDs check
- Subnet Manager check
- Partition keys check
- Nodes information

Return Codes

```
0 - Success
1 - Failure (with description)
```

9.4.1.1 Link Level Retransmission (LLR) in FDR Links

With the introduction of FDR 56 Gbps technology, Mellanox enabled a proprietary technology called LLR (Link Level Retransmission) to improve the reliability of FDR links.

This proprietary LLR technology adds additional CRC checking to the data stream and retransmits portions of packets with CRC errors at the local link level. Customers should be aware of the following facts associated with LLR technology:

- Traditional methods of checking the link health can be masked because the LLR technology automatically fixes errors. The traditional IB symbol error counter will show no errors when LLR is active.
- Latency of the fabric can be impacted slightly due to LLR retransmissions. Traditional IB performance utilities can be used to monitor any latency impact.
- Bandwidth of links can be reduced if cable performance degrades and LLR retransmissions become too numerous. Traditional IB bandwidth performance utilities can be used to monitor any bandwidth impact.

Due to these factors, an LLR retransmission rate counter has been added to the ibdiagnet utility that can give end users an indication of the link health.

➢ ***To monitor LLR retransmission rate:***

1. Run ibdiagnet, no special flags required.
2. If the LLR retransmission rate limit is exceeded it will print to the screen.

   The default limit is set to 500 and requires further investigation if exceeded.

The LLR retransmission rate is reflected in the results file `/var/tmp/ibdiagnet2/ibdiagnet2.pm`.

The default value of 500 retransmissions/sec has been determined by Mellanox based on the extensive simulations and testing. Links exhibiting a lower LLR retransmission rate should not raise special concern.

9.4.2 ibdiagnet (of ibutils) - IB Net Diagnostic

Please note that ibdiagnet is an obsolete package. We recommend using ibdiagnet from ibutils2.

This version of ibdiagnet is included in the ibutils package, and it is not run by default after installing Mellanox OFED.

To use this ibdiagnet version and not that of the ibutils package, you need to specify the full path: /opt/ibutils/bin

9.4.3 ibdiagpath - IB Diagnostic Path

ibdiagpath is located at: /opt/ibutisl/bin.

`ibdiagpath` traces a path between two end-points and provides information regarding the nodes and ports traversed along the path. It utilizes device specific health queries for the different devices along the path.

The way `ibdiagpath` operates depends on the addressing mode used on the command line. If directed route addressing is used (-d flag), the local node is the source node and the route to the destination port is known apriori. On the other hand, if LID-route (or by-name) addressing is employed, then the source and destination ports of a route are specified by their LIDs (or by the names defined in the topology file). In this case, the actual path from the local port to the source port, and from the source port to the destination port, is defined by means of Subnet Management Linear Forwarding Table queries of the switch nodes along that path. Therefore, the path cannot be predicted as it may change.

`ibdiagpath` should not be supplied with contradicting local ports by the -p and -d flags (see synopsis descriptions below). In other words, when ibdiagpath is provided with the options -p and -d together, the first port in the direct route must be equal to the one specified in the “-p” option. Otherwise, an error is reported.

When ibdiagpath queries for the performance counters along the path between the source and destination ports, it always traverses the LID route, even if a directed route is specified. If along the LID route one or more links are not in the ACTIVE state, ibdiagpath reports an error.

Moreover, the tool allows omitting the source node in LID-route addressing, in which case the local port on the machine running the tool is assumed to be the source.

Synopsis

```
ibdiagpath {-n <[src-name,]dst-name>|-l <[src-lid,]dst-lid>|-d <p1,p2,p3,...>} [-c <count>]
[-v] [-t <topo-file>] [-s <sys-name>] [-ic<dev-index>]c[-p <port-num>] [-o <out-dir>]
[-lw <1x|4x|12x>] [-ls <2.5|5|10>][-pm] [-pc] [-P <<PM counter>=<Trash Limit>>]
```

Options

-n <[src-name,]dst-name>	Names of the source and destination ports (as defined in the topology file; source may be omitted -> local port is assumed to be the source)
-l <[src-lid,]dst-lid>	Source and destination LIDs (source may be omitted --> the local port is assumed to be the source)
-d <p1,p2,p3,...>	Directed route from the local node (which is the source) and the destination node
-c <count>	The minimal number of packets to be sent across each link (default = 100)
-v	Enable verbose mode
-t <topo-file>	Specifies the topology file name
-s <sys-name>	Specifies the local system name. Meaningful only if a topology file is specified
-i <dev-index>	Specifies the index of the device of the port used to connect to the IB fabric (in case of multiple devices on the local system)
-p <port-num>	Specifies the local device's port number used to connect to the IB fabric
-o <out-dir>	Specifies the directory where the output files will be placed (default = /tmp)
-lw <1x|4x|12x>	Specifies the expected link width
-ls <2.5|5|10>	Specifies the expected link speed
-pm	Dump all the fabric links, pm Counters into ibdiagnet.pm
-pc	Reset all the fabric links pmCounters
-P <PM=<Trash>>	If any of the provided pm is greater then its provided value, print it to screen
-h|--help	Prints the help page information
-V|--version	Prints the version of the tool
--vars	Prints the tool's environment variables and their values

Output Files

Table 30 - ibdiagpath Output Files

Output File	Description
ibdiagpath.log	A dump of all the application reports generated according to the provided flags
ibdiagnet.pm	A dump of the Performance Counters values, of the fabric links

Error Codes

```
1 - The path traced is un-healthy
2 - Failed to parse command line options
3 - More then 64 hops are required for traversing the local port to the
    "Source" port and then to the "Destination" port
4 - Unable to traverse the LFT data from source to destination
5 - Failed to use Topology File
6 - Failed to load required Package
```

9.4.4 ibstat

ibstat is a binary which displays basic information obtained from the local IB driver. Output includes LID, SMLID, port state, link width active, and port physical state.

Synopsis

```
ibstat [-d(ebug)] [-l(ist_of_cas)] [-s(hort)] [-p(ort_list)] [-V(ersion)] [-h] <ca_name>
[portnum]
```

Options

The table below lists the various flags of the command.

Most OpenIB diagnostics take the following common flags. The exact list of supported flags per utility can be found in the usage message and can be shown using the util_name -h syntax..

Table 31 - ibstat Flags and Options

Flag	Description
-l, --list_of_cas	List all IB devices
-s, --short	Short output
-p, --port_list	Show port list
ca_name	InfiniBand device name
portnum	Port number of InfiniBand device
--debug, -d/-ddd/-d -d -d	Raise the IB debugging level
--help, -h	Show the usage message
-verbose, -v/-vv/-v -v -v	Increase the application verbosity level
--version, -V	Show the version info
--usage, -u	usage message

Examples

```
ibstat           # display status of all ports on all IB devices
ibstat -l        # list all IB devices
ibstat -p        # show port guids
ibstat mthca0 2  # show status of port 2 of 'mthca0'
```

9.4.5 ibtracert

ibtracert uses SMPs to trace the path from a source GID/LID to a destination GID/LID. Each hop along the path is displayed until the destination is reached or a hop does not respond. By using the -m option, multicast path tracing can be performed between source and destination nodes.

Synopsys

```
ibtracert [-d(ebug)] [-v(erbose)] [-D(irect)] [-L(id)] [-e(rrors)] [-u(sage)] [-G(uids)]
[-f(orce)] [-n(o_info)] [-m mlid] [-s smlid] [-C ca_name][-P ca_port] [-t(imeout) timeout_ms]
[-V(ersion)] [--node-name--map <node-name-map>] [-h(elp)] [<dest dr_path|lid|guid> [<startlid>
[<endlid>]]
```

Options

The table below lists the various flags of the command.

Most OpenIB diagnostics take the following common flags. The exact list of supported flags per utility can be found in the usage message and can be shown using the util_name -h syntax.

Table 32 - ibtracert Flags and Options

Flag	Description
--force, -f	Force
-n, --no_info	Simple format; do not show additional information
--mlid, -m <mlid>	Shows the multicast trace of the specified mlid
--node-name-map <node-name-map>	Specifies a node name map. The node name map file maps GUIDs to more user friendly names.
--debug, -d/-ddd/-d -d -d	Raises the IB debugging level
--Lid, -L	Uses LID address argument
--errors, -e	Shows send and receive errors
--usage, -u	Usage message
--Guid, -G	Uses GUID address argument. In most cases, it is the Port GUID. Example: "0x08f1040023"
--sm_port, -s <smlid>	Uses 'smlid' as the target lid for SM/SA queries
--help, -h	Shows the usage message
-verbose, -v/-vv/-v -v -v	Increases the application verbosity level
--version, -V	Shows the version info
--Ca, -C <ca_name>	Uses the specified ca_name
--Port, -P <ca_port>	Uses the specified ca_port
--timeout, -t <timeout_ms>	Overrides the default timeout for the solicited mads

Examples

- Unicast examples

```
ibtracert 4 16              # show path between lids 4 and 16
ibtracert -n 4 16           # same, but using simple output format
ibtracert -G 0x8f1040396522d 0x002c9000100d051  # use guid addresses
```

- Multicast example

```
ibtracert -m 0xc000 4 16    # show multicast path of mlid 0xc000 between
lids 4 and 16
```

9.4.6 ibqueryerrors

The default behavior is to report the port error counters which exceed a threshold for each port in the fabric. The default threshold is zero (0). Error fields can also be suppressed entirely.

In addition to reporting errors on every port, ibqueryerrors can report the port transmit and receive data as well as report full link information to the remote port if available.

Synopsis

```
ibqueryerrors [options]
```

Options

Table 33 - ibqueryerrors Flags and Options

Flags	Description
XmitWait	The amount of time it takes for the port to be selected. PortSelect had data to transmit, but no data was sent during the entire wait time because of insufficient credits or lack of arbitration. Usually, it identifies slow receivers or traffic oversubscription on particular links. The counter is only indicative (not an error)
-s <err1,err2,...>	Suppresses the errors listed in the comma separated list provided. For example: To suppress XmitWait counter in the output, run `ibqueryerrors -s PortXmitWait`
-c	Suppresses some of the common "side effect" counters. These counters usually do not indicate an error condition and can be usually be safely ignored. **Note:** Currently only PortSwitchRcvRelayErrors counter can be suppressed.
-G <port_guid> -S <port_guid> --port-guid	Report results for the port specified. For switches results are printed for all ports not just switch port 0.
-S same as "-G"	Provided only for backward compatibility
-D <direct_route>	Reports results for the port specified. For switches results are printed for all ports not just switch port 0.
-r	Reports the port information. This includes LID, port, external port (if applicable), link speed setting, remote GUID, remote port, remote external port (if applicable), and remote node description information.
--data	Includes the optional transmit and receive data counters.
--skip-sl	Uses the default sl for queries. This is not recommended when using a QoS aware routing engine as it can cause a credit deadlock.
--threshold-file	Specifies an alternate threshold file. The default is: /opt/ufm/files/conf/infiniband-diags/error_thresholds

Table 33 - ibqueryerrors Flags and Options

Flags	Description
--switch	Prints data for switches only.
--ca	Prints data for CA's only.
--router	Prints data for routers only
--clear-errors-k	Clear error counters after read. -k and -K can be used together to clear both errors and counters.
--clear-counts -K	Clear data counters after read. CAUTION: clearing data counters will occur regardless of if they are printed or not. This is because data counters are only printed on ports which have errors. This means if a port has 0 errors and the -K option is specified the data counters will be cleared without any printed output.
-details	Includes receive error and transmits discard details
--load-cache <filename>	Loads and uses the cached ibnetdiscover data stored in the specified filename. May be useful for outputting and learning about other fabrics or a previous state of a fabric. Cannot be used if user specifies a direct route path. See ibnetdiscover for information on caching ibnetdiscover output.
-R	This option is obsolete (and has no effect).
-d	Raises the IB debugging level. May be used several times (-ddd or -d -d -d).
-e	Shows send and receive errors (time-outs and others)
-h	Shows the usage message
-v	Increases the application verbosity level. May be used several times (-vv or -v -v -v)
-C <ca_name>	Uses the specified ca_name.
-P <ca_port>	Uses the specified ca_port.
-t <timeout_ms>	Overrides the default timeout for the solicited mads.

Exit Status

If a failure to scan the fabric occurs return -1. If the scan succeeds without errors beyond thresholds return 0. If errors are found on ports beyond thresholds return 1.

Files

/etc/infiniband-diags/error_threshold

Define threshold values for errors. File format is simple "name=val".

Comments begin with '#'

Example:

```
# Define thresholds for error counters
                    SymbolErrorCounter=10
                    LinkErrorRecoveryCounter=10
                    VL15Dropped=100
```

9.4.7 iblinkinfo

iblinkinfo reports link info for each port in an IB fabric, node by node. Optionally, iblinkinfo can do partial scans and limit its output to parts of a fabric.

9.4.7.1 iblinkinfo Synopsis

```
[-hcdl -C <ca_name> -P <ca_port> -p -S <port_guid> -G <port_guid> -D <direct_route> --load-cache
<filename>]
```

9.4.7.2 iblinkinfo Flags and Options

Table 34 - iblinkinfo Flags and Options

Flags	Description
-S <port_guid> -G <port_guid> --port-guid	Starts partial scan at the port specified by <port_guid> (hex format)
-D <direct_route>	Starts partial scan at the port specified by the direct route path.
-l	Prints all information for each link on one line. Default is to print a header with the node information and then a list for each port (useful for grep´ing output).
-d	Prints only nodes which have a port in the "Down" state.
-p	Prints additional port settings (<Life-Time>,<HoqLife>,<VLStall-Count>)
-C <ca_name>	Uses the specified ca_name for the search.
-P <ca_port>	Uses the specified ca_port for the search.
-R	(This option is obsolete and does nothing)

Table 34 - iblinkinfo Flags and Options

Flags	Description
--load-cache <filename>	Loads and use the cached ibnetdiscover data stored in the specified filename. May be useful for outputting and learning about other fabrics or a previous state of a fabric. Cannot be used if user specifies a direct route path. See ibnetdiscover for information on caching ibnetdiscover output.
--diff <filename>	Loads cached ibnetdiscover data and do a diff comparison to the current network or another cache. A special diff output for iblinkinfo output will be displayed showing differences between the old and current fabric links. Be default, the following are compared for differences: port connections and port state. See ibnetdiscover for information on caching ibnetdiscover output.
--diffcheck <key(s)>	Specifies what diff checks should be done in the--diffoption above. Comma separate multiple diff check key(s). The available diff checks are:port = port connections,state = port state, lid = lids, nodedesc = node descriptions. If port is specified alongside lid or nodedesc, remote port lids and node descriptions will also be compared.
--filterdownports <filename>	Filters downports indicated in a ibnetdiscover cache. If a port was previously indicated as down in the specified cache, and is still down, do not output it in the resulting output. This option may be particularly useful for environments where switches are not fully populated, thus much of the default iblinkinfo info is considered un-useful. See ibnetdiscover for information on caching ibnetdiscover output.

9.4.8 saquery

saquery issues the selected SA query. Node records are queried by default.

9.4.8.1 saquery Synopsis

```
saquery [-h] [-d] [-p] [-N] [--list | -D] [-S] [-I] [-L] [-l] [-G] [-O] [-U] [-c] [-s] [-g] [-m]
[-x] [-C ca_name] [-P ca_port][--smkey  val][-t(imeout) <msec>] [--src-to-dst <src:dst>]
[--sgid-to-dgid <sgid-dgid>] [--node-name-map <node-name-map>] [<name> | <lid> | <guid>]
```

9.4.8.2 saquery Options

Table 35 - saquery Flags and Options

Flags	Description
-p	Gets PathRecord info.

Table 35 - saquery Flags and Options

Flags	Description
-N	Gets NodeRecord info.
--list | -D	Gets NodeDescriptions of CAs only.
-S	Gets ServiceRecord info.
-I	Gets InformInfoRecord (subscription) info.
-L	Returns the Lids of the name specified
-l	Returns the unique Lid of the name specified
-G	Returns the Guids of the name specified
-O	Returns the name for the Lid specified
-U	Returns the name for the Guid specified
-C	Gets the SA's class port info
-s	Returns the PortInfoRecords with isSM or isSMdisabled capability mask bit on.
-g	Gets multicast group info
-m	Gets multicast member info. If a group is specified, limit the output to the group specified and print one line containing only the GUID and node description for each entry. Example: saquery -m 0xc000
-x	Gets LinkRecord info.
--src-to-dst	Gets a PathRecord for <src:dst> where src and dst are either node names or LIDs.
--sgid-to-dgid	Gets a PathRecord for sgid to dgid where both GIDs are in an IPv6 format acceptable to inet_pton(3).
-C <ca_name>	Uses the specified ca_name.
-P <ca_port>	Uses the specified ca_port.
--smkey <val>	Uses SM_Key value for the query. Will be used only with "trusted" queries. If non-numeric value (like'x') is specified then saquery will prompt for a value.
-t, -timeout <msec>	Specifies SA query response timeout in milliseconds. Default is 100 milliseconds. You may want to use this option if IB_TIMEOUT is indicated.

Table 35 - saquery Flags and Options

Flags	Description
--node-name-map <node-name-map>	Specifies a node name map.The node name map file maps GUIDs to more user friendly names. See ibnetdiscover(8) for node name map file format.Only used with the -O and -U options. • Supported query names (and aliases): ClassPortInfo (CPI) NodeRecord (NR) [lid] PortInfoRecord (PIR) [[lid]/[port]/ [options]] SL2VLTableRecord (SL2VL) [[lid]/ [in_port]/[out_port]] PKeyTableRecord (PKTR) [[lid]/[port]/ [block]] VLArbitrationTableRecord (VLAR) [[lid]/[port]/[block]] InformInfoRecord (IIR) LinkRecord (LR) [[from_lid]/ [from_port]] [[to_lid]/[to_port]] ServiceRecord (SR) PathRecord (PR) MCMemberRecord (MCMR) LFTRecord (LFTR) [[lid]/[block]] MFTRecord (MFTR) [[mlid]/ [position]/[block]] GUIDInfoRecord (GIR) [[lid]/[block]]
-d	enables debugging.
-h	Shows help.

9.4.9 smpdump

smpdump is a general purpose SMP utility which gets SM attributes from a specified SMA. The result is dumped in hex by default.

9.4.9.1 smpdump Synopsis

```
smpdump [-s(ring)]  [-D(irect)] [-C ca_name] [-P ca_port] [-t(imeout) timeout_ms] [-V(ersion)]
[-h(elp)] <dlid|dr_path> <attr> [mod]
```

9.4.9.2 smpdump Options

Table 36 - smpdump Flags and Options

Flags	Description
attr	IBA attribute ID for SM attribute
mod	IBA modifier for SM attribute
Debugging Flags	**Description**
NOTE: Most OpenIB diagnostics take the following common flags. The exact list of supported flags per utility can be found in the usage message and can be shown using the util_name -h syntax.	

Table 36 - smpdump Flags and Options

Flags	Description
-d	Raises the IB debugging level. Can be used several times (-ddd or -d -d -d).
-e	Shows send and receive errors (timeouts and others)
-h	Shows the usage message
-v	Increases the application verbosity level. Can be used several times (-vv or -v -v -v)
-V	Shows the version info.
Addressing Flags	**Description**
-D	Uses directed path address arguments. The path is a comma separated list of out ports. Examples: "0" # self port "0,1,2,1,4" # out via port 1, then 2, ...
-G	Uses GUID address argument. In most cases, it is the Port GUID. Example: "0x08f1040023"
-s <smlid>	Uses 'smlid' as the target lid for SM/SA queries.
Flags	**Description**
-C <ca_name>	Uses the specified ca_name.
-P <ca_port>	Uses the specified ca_port.
-t <timeout_ms>	Overrides the default timeout for the solicited mads.

9.4.9.3 Multiple CA/Multiple Port Support

When no IB device or port is specified, the port to use is selected by the following criteria:

1. The first port that is ACTIVE.
2. If not found, the first port that is UP (physical link up).

If a port and/or CA name is specified, the user request is attempted to be fulfilled, and will fail if it is not possible.

Examples

Direct Routed Examples:

```
smpdump -D 0,1,2,3,5 16 # NODE DESC
smpdump -D 0,1,2 0x15 2 # PORT INFO, port 2
```

LID Routed Examples:

```
smpdump 3 0x15 2         # PORT INFO, lid 3 port 2
smpdump 0xa0 0x11       # NODE INFO, lid 0xa0
```

9.4.10 ibv_devices

Lists InfiniBand devices available for use from userspace, including node GUIDs.

Synopsis

```
ibv_devices
```

Examples

1. List the names of all available InfiniBand devices.

```
> ibv_devices
    device                 node GUID
    ------              ----------------
    mthca0              0002c9000101d150
    mlx4_0              0000000000073895
```

9.4.11 ibv_devinfo

Queries InfiniBand devices and prints about them information that is available for use from user-space.

Synopsis

```
ibv_devinfo [-d <device>] [-i <port>] [-l] [-v]
```

Output Files

Table 37 lists the various flags of the command.

Table 37 - ibv_devinfo Flags and Options

Flag	Optional / Mandatory	Default (If Not Specified)	Description
-d <device> --ib-dev=<device>	Optional	First found device	Run the command for the provided IB device ‘device’
-i <port> --ib-port=<port>	Optional	All device ports	Query the specified device port <port>
-l --list	Optional	Inactive	Only list the names of InfiniBand devices
-v --verbose	Optional	Inactive	Print all available information about the InfiniBand device(s)

Examples

1. List the names of all available InfiniBand devices.

```
> ibv_devinfo -l
2 HCAs found:
```

```
        mthca0
        mlx4_0
```

2. Query the device mlx4_0 and print user-available information for its Port 2.

```
> ibv_devinfo -d mlx4_0 -i 2
hca_id: mlx4_0
        fw_ver:                         2.5.944
        node_guid:                      0000:0000:0007:3895
        sys_image_guid:                 0000:0000:0007:3898
        vendor_id:                      0x02c9
        vendor_part_id:                 25418
        hw_ver:                         0xA0
        board_id:                       MT_04A0140005
        phys_port_cnt:                  2
                port:   2
                        state:                  PORT_ACTIVE (4)
                        max_mtu:                2048 (4)
                        active_mtu:             2048 (4)
                        sm_lid:                 1
                        port_lid:               1
                        port_lmc:               0x00
```

9.4.12 ibdev2netdev

ibdev2netdev enables association between IB devices and ports and the associated net device. Additionally it reports the state of the net device link.

Synopsys

```
ibdev2netdev [-v] [-h]
```

Options

```
-v Enable verbose mode. Adds additional information such as: Device ID, Part Number,
Card Name, Firmware version, IB port state.
-h Print help messages.
```

Example:

```
sw417:~/BXOFED-1.5.2-20101128-1524 # ibdev2netdev -v
mlx4_0 (MT26428 - MT1006X00034) FALCON QDR      fw 2.7.9288 port 1 (ACTIVE) ==> eth5
(Down)
mlx4_0 (MT26428 - MT1006X00034) FALCON QDR      fw 2.7.9288 port 1 (ACTIVE) ==> ib0
(Down)
mlx4_0 (MT26428 - MT1006X00034) FALCON QDR      fw 2.7.9288 port 2 (DOWN  ) ==> ib1
(Down)
mlx4_1 (MT26448 - MT1023X00777) Hawk Dual Port  fw 2.7.9400 port 1 (DOWN  ) ==> eth2
(Down)
```

```
mlx4_1 (MT26448 - MT1023X00777) Hawk Dual Port  fw 2.7.9400 port 2 (DOWN  ) ==> eth3
(Down)
sw417:~/BXOFED-1.5.2-20101128-1524 # ibdev2netdev
mlx4_0 port 1 ==> eth5 (Down)
mlx4_0 port 1 ==> ib0 (Down)
mlx4_0 port 2 ==> ib1 (Down)
mlx4_1 port 1 ==> eth2 (Down)
mlx4_1 port 2 ==> eth3 (Down)
```

9.4.13 ibstatus

Displays basic information obtained from the local InfiniBand driver. Output includes LID, SMLID, port state, port physical state, port width and port rate.

Synopsis

```
ibstatus  [-h] [<device name>[:<port>]]*
```

Output Files

Table 38 lists the various flags of the command.

Table 38 - ibstatus Flags and Options

Flag	Optional / Mandatory	Default (If Not Specified)	Description
-h	Optional		Print the help menu
<device>	Optional	All devices	Print information for the specified device. May specify more than one device
<port>	Optional, but requires specifying a device name	All ports of the specified device	Print information for the specified port only (of the specified device)

Examples

1. List the status of all available InfiniBand devices and their ports.

```
> ibstatus
Infiniband device 'mlx4_0' port 1 status:
        default gid:     fe80:0000:0000:0000:0000:0000:0007:3896
        base lid:        0x3
        sm lid:          0x3
        state:           4: ACTIVE
        phys state:      5: LinkUp
        rate:            20 Gb/sec (4X DDR)

Infiniband device 'mlx4_0' port 2 status:
```

```
        default gid:     fe80:0000:0000:0000:0000:0000:0007:3897
        base lid:        0x1
        sm lid:          0x1
        state:           4: ACTIVE
        phys state:      5: LinkUp
        rate:            20 Gb/sec (4X DDR)

Infiniband device 'mthca0' port 1 status:
        default gid:     fe80:0000:0000:0000:0002:c900:0101:d151
        base lid:        0x0
        sm lid:          0x0
        state:           2: INIT
        phys state:      5: LinkUp
        rate:            10 Gb/sec (4X)

Infiniband device 'mthca0' port 2 status:
        default gid:     fe80:0000:0000:0000:0002:c900:0101:d152
        base lid:        0x0
        sm lid:          0x0
        state:           2: INIT
        phys state:      5: LinkUp
        rate:            10 Gb/sec (4X)
```

2. List the status of specific ports of specific devices.

```
> ibstatus mthca0:1 mlx4_0:2
Infiniband device 'mthca0' port 1 status:
        default gid:     fe80:0000:0000:0000:0002:c900:0101:d151
        base lid:        0x0
        sm lid:          0x0
        state:           2: INIT
        phys state:      5: LinkUp
        rate:            10 Gb/sec (4X)

Infiniband device 'mlx4_0' port 2 status:
        default gid:     fe80:0000:0000:0000:0000:0000:0007:3897
        base lid:        0x1
        sm lid:          0x1
        state:           4: ACTIVE
        phys state:      5: LinkUp
        rate:            20 Gb/sec (4X DDR)
```

9.4.14 ibportstate

Enables querying the logical (link) and physical port states of an InfiniBand port. It also allows adjusting the link speed that is enabled on any InfiniBand port.

If the queried port is a *switch* port, then ibportstate can be used to

- disable, enable or reset the port
- validate the port's link width and speed against the peer port

Synopsis

```
ibportstate [options] <dest dr_path|lid|guid> <portnum> [<op>]
```

Output Files

```
-L, --Lid                       The address specified is a LID
-G, --Guid                      The address specified is a Port GUID
-D, --Direct                    The address specified is a directed route
-s, --sm_port <smlid>           Use 'smlid' as the target lid for SA queries.
-C, --Ca <ca_name>              use the specified ca_name.
-P, --Port <ca_port>            Use the specified ca_port:
                                Local port Selection
                                Multiple port/Multiple CA support: when no IB device or port
                                is specified (see the "local umad parameters" below), the
                                libibumad library selects the port to use by the following
                                criteria:
                                1. the first port that is ACTIVE.
                                2. if not found, the first port that is UP (physical link
                                up).
                                If a port and/or CA name is specified, the libibumad library
                                attempts to fulfill the user request, and will fail if it is
                                not possible.
--config, -z  <config_file>     Specify alternate config file. Default: /etc/infiniband-
                                diags/ibdiag.conf
-t, --timeout <timeout_ms>      override the default timeout for the solicited mads.
-y, --m_key <key>               Use the specified M_key for requests. If non-numeric value
                                (like 'x') is specified then a value will be prompted for.
-h, --help                      show the usage message
-d                              Raise the IB debugging level.  May be used several times (-
                                ddd or -d -d -d).
-e                              Show send and receive errors (timeouts and others)
-K, --show_keys                 Show security keys (mkey, smkey, etc.) associated with the
                                request.
-v, --verbose                   Increase the application verbosity level. May be used sev-
                                eral times (-vv or -v -v -v)
-V, --version                   Show the version info.
```

In case of multiple channel adapters (CAs) or multiple ports without a CA/port being specified, a port is chosen by the utility according to the following criteria:

1. The first ACTIVE port that is found.
2. If not found, the first port that is UP (physical link state is LinkUp).

Examples

1. Query the status of Port 1 of CA mlx4_0 (using ibstatus) and use its output (the LID – 3 in this case) to obtain additional link information using ibportstate.

```
>  ibstatus mlx4_0:1
Infiniband device 'mlx4_0' port 1 status:
        default gid:     fe80:0000:0000:0000:0000:0000:9289:3895
        base lid:        0x3
        sm lid:          0x3
        state:           2: INIT
        phys state:      5: LinkUp
        rate:            20 Gb/sec (4X DDR)

> ibportstate -C mlx4_0 3 1 query
PortInfo:
# Port info: Lid 3 port 1
LinkState:.....................Initialize
PhysLinkState:.................LinkUp
LinkWidthSupported:............1X or 4X
LinkWidthEnabled:..............1X or 4X
LinkWidthActive:...............4X
LinkSpeedSupported:............2.5 Gbps or 5.0 Gbps
LinkSpeedEnabled:..............2.5 Gbps or 5.0 Gbps
LinkSpeedActive:...............5.0 Gbps
```

2. Query the status of two channel adapters using directed paths.

```
>  ibportstate -C mlx4_0 -D 0 1
PortInfo:
# Port info: DR path slid 65535; dlid 65535; 0 port 1
LinkState:.....................Initialize
PhysLinkState:.................LinkUp
LinkWidthSupported:............1X or 4X
LinkWidthEnabled:..............1X or 4X
LinkWidthActive:...............4X
LinkSpeedSupported:............2.5 Gbps or 5.0 Gbps
LinkSpeedEnabled:..............2.5 Gbps or 5.0 Gbps
LinkSpeedActive:...............5.0 Gbps

>  ibportstate -C mthca0 -D 0 1
PortInfo:
# Port info: DR path slid 65535; dlid 65535; 0 port 1
```

```
LinkState:.....................Down
PhysLinkState:.................Polling
LinkWidthSupported:............1X or 4X
LinkWidthEnabled:..............1X or 4X
LinkWidthActive:...............4X
LinkSpeedSupported:............2.5 Gbps
LinkSpeedEnabled:..............2.5 Gbps
LinkSpeedActive:...............2.5 Gbps
```

3. Change the speed of a port.

```
# First query for current configuration
>  ibportstate -C mlx4_0 -D 0 1
PortInfo:
# Port info: DR path slid 65535; dlid 65535; 0 port 1
LinkState:.....................Initialize
PhysLinkState:.................LinkUp
LinkWidthSupported:............1X or 4X
LinkWidthEnabled:..............1X or 4X
LinkWidthActive:...............4X
LinkSpeedSupported:............2.5 Gbps or 5.0 Gbps
LinkSpeedEnabled:..............2.5 Gbps or 5.0 Gbps
LinkSpeedActive:...............5.0 Gbps

# Now change the enabled link speed
>  ibportstate -C mlx4_0 -D 0 1 speed 2
ibportstate -C mlx4_0 -D 0 1 speed 2
Initial PortInfo:
# Port info: DR path slid 65535; dlid 65535; 0 port 1
LinkSpeedEnabled:..............2.5 Gbps

After PortInfo set:
# Port info: DR path slid 65535; dlid 65535; 0 port 1
LinkSpeedEnabled:..............5.0 Gbps (IBA extension)

# Show the new configuration
>  ibportstate -C mlx4_0 -D 0 1
PortInfo:
# Port info: DR path slid 65535; dlid 65535; 0 port 1
LinkState:.....................Initialize
PhysLinkState:.................LinkUp
LinkWidthSupported:............1X or 4X
LinkWidthEnabled:..............1X or 4X
LinkWidthActive:...............4X
LinkSpeedSupported:............2.5 Gbps or 5.0 Gbps
LinkSpeedEnabled:..............5.0 Gbps (IBA extension)
LinkSpeedActive:...............5.0 Gbps
```

9.4.15 ibroute

Uses SMPs to display the forwarding tables—unicast (LinearForwardingTable or LFT) or multicast (MulticastForwardingTable or MFT)—for the specified switch LID and the optional lid (mlid) range. The default range is all valid entries in the range 1 to FDBTop.

Synopsis

```
ibroute [-h] [-d] [-v] [-V] [-a] [-n] [-D] [-G] [-M] [-s <smlid>] \ [-C <ca_name>]
[-P <ca_port>] [-t <timeout_ms>] \ [<dest dr_path|lid|guid> [<startlid> [<endlid>]]]
```

Output Files

Table 39 lists the various flags of the command.

Table 39 - ibportstate Flags and Options

Flag	Optional / Mandatory	Default (If Not Specified)	Description
-h(help)	Optional		Print the help menu
-d(ebug)	Optional		Raise the IB debug level. May be used several times for higher debug levels (-ddd or -d -d -d)
-a(ll)	Optional		Show all LIDs in range, including invalid entries
-v(erbose)	Optional		Increase verbosity level. May be used several times for additional verbosity (-vvv or -v -v -v)
-V(ersion)	Optional		Show version info
-a(ll)	Optional		Show all LIDs in range, including invalid entries
-n(o_dests)	Optional		Do not try to resolve destinations
-D(irect)	Optional		Use directed path address arguments. The path is a comma separated list of out ports. Examples: ‘0’ – self port ‘0,1,2,1,4’ – out via port 1, then 2, ...
-G(uid)	Optional		Use GUID address argument. In most cases, it is the Port GUID. Example: ‘0x08f1040023’
-M(ulticast)	Optional		Show multicast forwarding tables. The parameters <startlid> and <endlid> specify the MLID range.
-s <smlid>	Optional		Use <smlid> as the target LID for SM/SA queries
-C <ca_name>	Optional		Use the specified channel adapter or router

Table 39 - ibportstate Flags and Options

Flag	Optional / Mandatory	Default (If Not Specified)	Description
-P <ca_port>	Optional		Use the specified port
-t <timeout_ms>	Optional		Override the default timeout for the solicited MADs [msec]
<dest dr_path | lid | guid>	Optional		Destination's directed path, LID, or GUID
<startlid>	Optional		Starting LID in an MLID range
<endlid>	Optional		Ending LID in an MLID range

Examples

1. Dump all Lids with valid out ports of the switch with Lid 2.

```
> ibroute 2
Unicast lids [0x0-0x8] of switch Lid 2 guid 0x0002c902fffff00a (MT47396 Infiniscale-III
Mellanox Technologies):
  Lid  Out   Destination
       Port     Info
0x0002 000 : (Switch portguid 0x0002c902fffff00a: 'MT47396 Infiniscale-III Mellanox
Technologies')
0x0003 021 : (Switch portguid 0x000b8cffff004016: 'MT47396 Infiniscale-III Mellanox
Technologies')
0x0006 007 : (Channel Adapter portguid 0x0002c90300001039: 'sw137 HCA-1')
0x0007 021 : (Channel Adapter portguid 0x0002c9020025874a: 'sw157 HCA-1')
0x0008 008 : (Channel Adapter portguid 0x0002c902002582cd: 'sw136 HCA-1')
5 valid lids dumped
```

2. Dump all Lids with valid out ports of the switch with Lid 2.

```
> ibroute 2
Unicast lids [0x0-0x8] of switch Lid 2 guid 0x0002c902fffff00a (MT47396 Infiniscale-III
Mellanox Technologies):
  Lid  Out   Destination
       Port     Info
0x0002 000 : (Switch portguid 0x0002c902fffff00a: 'MT47396 Infiniscale-III Mellanox
Technologies')
0x0003 021 : (Switch portguid 0x000b8cffff004016: 'MT47396 Infiniscale-III Mellanox
Technologies')
0x0006 007 : (Channel Adapter portguid 0x0002c90300001039: 'sw137 HCA-1')
0x0007 021 : (Channel Adapter portguid 0x0002c9020025874a: 'sw157 HCA-1')
0x0008 008 : (Channel Adapter portguid 0x0002c902002582cd: 'sw136 HCA-1')
5 valid lids dumped
```

3. Dump all Lids in the range 3 to 7 with valid out ports of the switch with Lid 2.

```
> ibroute 2 3 7
Unicast lids [0x3-0x7] of switch Lid 2 guid 0x0002c902fffff00a (MT47396 Infiniscale-III
Mellanox Technologies):
  Lid  Out   Destination
       Port     Info
0x0003 021 : (Switch portguid 0x000b8cffff004016: 'MT47396 Infiniscale-III Mellanox
Technologies')
0x0006 007 : (Channel Adapter portguid 0x0002c90300001039: 'sw137 HCA-1')
0x0007 021 : (Channel Adapter portguid 0x0002c9020025874a: 'sw157 HCA-1')
3 valid lids dumped
```

4. Dump all Lids with valid out ports of the switch with portguid `0x000b8cffff004016`.

```
> ibroute -G 0x000b8cffff004016
Unicast lids [0x0-0x8] of switch Lid 3 guid 0x000b8cffff004016 (MT47396 Infiniscale-III
Mellanox Technologies):
  Lid  Out   Destination
       Port     Info
0x0002 023 : (Switch portguid 0x0002c902fffff00a: 'MT47396 Infiniscale-III Mellanox
Technologies')
0x0003 000 : (Switch portguid 0x000b8cffff004016: 'MT47396 Infiniscale-III Mellanox
Technologies')
0x0006 023 : (Channel Adapter portguid 0x0002c90300001039: 'sw137 HCA-1')
0x0007 020 : (Channel Adapter portguid 0x0002c9020025874a: 'sw157 HCA-1')
0x0008 024 : (Channel Adapter portguid 0x0002c902002582cd: 'sw136 HCA-1')
5 valid lids dumped
```

5. Dump all non-empty mlids of switch with Lid 3.

```
> ibroute -M 3
Multicast mlids [0xc000-0xc3ff] of switch Lid 3 guid 0x000b8cffff004016 (MT47396
Infiniscale-III Mellanox Technologies):
            0                   1                   2
     Ports: 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4
 MLid
0xc000                                              x
0xc001                                              x
0xc002                                              x
0xc003                                              x
0xc020                 x
0xc021                 x
0xc022                 x
0xc023                 x
0xc024                 x
0xc040                 x
0xc041                 x
0xc042                 x
12 valid mlids dumped
```

9.4.16 smpquery

Provides a basic subset of standard SMP queries to query Subnet management attributes such as node info, node description, switch info, and port info.

Synopsis

```
smpquery [-h] [-d] [-e] [-v] [-D] [-G] [-s <smlid>] [-V] [-C <ca_name>] [-P <ca_port>] [-t
<timeout_ms>] [--node-name-map <node-name-map>] <op> <dest dr_path|lid|guid> [op params]
```

Output Files

Table 40 lists the various flags of the command.

Table 40 - smpquery Flags and Options

Flag	Optional / Mandatory	Default (If Not Specified)	Description
-h(help)	Optional		Print the help menu
-d(ebug)	Optional		Raise the IB debug level. May be used several times for higher debug levels (-ddd or -d -d -d)
-e(rr_show)	Optional		Show send and receive errors (timeouts and others)
-v(erbose)	Optional		Increase verbosity level. May be used several times for additional verbosity (-vvv or -v -v -v)
-D(irect)	Optional		Use directed path address arguments. The path is a comma separated list of out ports. Examples: ‘0’ – self port ‘0,1,2,1,4’ – out via port 1, then 2, ...
-G(uid)	Optional		Use GUID address argument. In most cases, it is the Port GUID. Example: ‘0x08f1040023’
-s <smlid>	Optional		Use <smlid> as the target LID for SM/SA queries
-V(ersion)	Optional		Show version info
-C <ca_name>	Optional		Use the specified channel adapter or router
-P <ca_port>	Optional		Use the specified port
-t <timeout_ms>	Optional		Override the default timeout for the solicited MADs [msec]

Table 40 - smpquery Flags and Options

Flag	Optional / Mandatory	Default (If Not Specified)	Description
<op>	Mandatory		Supported operations: nodeinfo <addr> nodedesc <addr> portinfo <addr> [<portnum>] switchinfo <addr> pkeys <addr> [<portnum>] sl2vl <addr> [<portnum>] vlarb <addr> [<portnum>] guids <addr> mepi <addr> [<portnum>]
<dest dr_path | lid | guid>	Optional		Destination's directed path, LID, or GUID

Examples

1. Query PortInfo by LID, with port modifier.

```
> smpquery portinfo 1 1
# Port info: Lid 1 port 1
Mkey:.........................0x0000000000000000
GidPrefix:....................0xfe80000000000000
Lid:..........................0x0001
SMLid:........................0x0001
CapMask:......................0x251086a
                              IsSM
                              IsTrapSupported
                              IsAutomaticMigrationSupported
                              IsSLMappingSupported
                              IsSystemImageGUIDsupported
                              IsCommunicatonManagementSupported
                              IsVendorClassSupported
                              IsCapabilityMaskNoticeSupported
                              IsClientRegistrationSupported
DiagCode:.....................0x0000
MkeyLeasePeriod:..............0
LocalPort:....................1
LinkWidthEnabled:.............1X or 4X
LinkWidthSupported:...........1X or 4X
LinkWidthActive:..............4X
LinkSpeedSupported:...........2.5 Gbps or 5.0 Gbps
LinkState:....................Active
PhysLinkState:................LinkUp
LinkDownDefState:.............Polling
```

```
ProtectBits:.....................0
LMC:.............................0
LinkSpeedActive:.................5.0 Gbps
LinkSpeedEnabled:................2.5 Gbps or 5.0 Gbps
NeighborMTU:.....................2048
SMSL:............................0
VLCap:...........................VL0-7
InitType:........................0x00
VLHighLimit:.....................4
VLArbHighCap:....................8
VLArbLowCap:.....................8
InitReply:.......................0x00
MtuCap:..........................2048
VLStallCount:....................0
HoqLife:.........................31
OperVLs:.........................VL0-3
PartEnforceInb:..................0
PartEnforceOutb:.................0
FilterRawInb:....................0
FilterRawOutb:...................0
MkeyViolations:..................0
PkeyViolations:..................0
QkeyViolations:..................0
GuidCap:.........................128
ClientReregister:................0
SubnetTimeout:...................18
RespTimeVal:.....................16
LocalPhysErr:....................8
OverrunErr:......................8
MaxCreditHint:...................0
RoundTrip:.......................0
```

2. Query SwitchInfo by GUID.

```
> smpquery -G switchinfo 0x000b8cffff004016
# Switch info: Lid 3
LinearFdbCap:....................49152
RandomFdbCap:....................0
McastFdbCap:.....................1024
LinearFdbTop:....................8
DefPort:.........................0
DefMcastPrimPort:................0
DefMcastNotPrimPort:.............0
LifeTime:........................18
StateChange:.....................0
LidsPerPort:.....................0
PartEnforceCap:..................32
```

```
InboundPartEnf:.................1
OutboundPartEnf:................1
FilterRawInbound:...............1
FilterRawOutbound:..............1
EnhancedPort0:..................0
```

3. Query NodeInfo by direct route.

```
> smpquery -D nodeinfo 0
# Node info: DR path slid 65535; dlid 65535; 0
BaseVers:.......................1
ClassVers:......................1
NodeType:.......................Channel Adapter
NumPorts:.......................2
SystemGuid:.....................0x0002c9030000103b
Guid:...........................0x0002c90300001038
PortGuid:.......................0x0002c90300001039
PartCap:........................128
DevId:..........................0x634a
Revision:.......................0x000000a0
LocalPort:......................1
VendorId:.......................0x0002c9
```

9.4.17 perfquery

Queries InfiniBand ports' performance and error counters. Optionally, it displays aggregated counters for all ports of a node. It can also reset counters after reading them or simply reset them.

Synopsis

```
perfquery [options]  [<lid|guid> [[port(s)] [reset_mask]]]
```

Output Files

Table 41 lists the various flags of the command.

Table 41 - perfquery Options

Flag	Description
--extended, -x	Show extended port counters
--xmtsl, -X	Show Xmt SL port counters
--rcvsl, -S	Show Rcv SL port counters
--xmtdisc, -D	Show Xmt Discard Details
--rcverr, -E	Show Rcv Error Details
--extended_speeds, -T	Show port extended speeds counters
--oprcvcounters	Show Rcv Counters per Op code

Table 41 - perfquery Options

Flag	Description
--flowctlcounters	Show flow control counters
--vloppackets	Show packets received per Op code per VL
--vlopdata	Show data received per Op code per VL
--vlxmitflowctlerrors	Show flow control update errors per VL
--vlxmitcounters	Show ticks waiting to transmit counters per VL
--swportvlcong	Show sw port VL congestion
--rcvcc	Show Rcv congestion control counters
--slrcvfecn	Show SL Rcv FECN counters
--slrcvbecn	Show SL Rcv BECN counters
--xmitcc	Show Xmit congestion control counters
--vlxmittimecc	Show VL Xmit Time congestion control counters
--smplctl, -c	Show samples control
--all_ports, -a	Show aggregated counters
--loop_ports, -l	Iterate through each port
--reset_after_read, -r	Reset counters after read
--Reset_only, -R	Only reset counters
--config, -z <config>	Use config file, default: /etc/infiniband-diags/ibdiag.conf
--Ca, -C <ca>	Ca name to Use
--Port, -P <port>	Ca port number to Use
--Lid, -L	Use LID address argument
--Guid, -G	use GUID address argument
--timeout, -t <ms>	Timeout in ms
--sm_port, -s	<lid> SM port lid
--m_key, -y <key>	M_Key to use in request
--errors, -e	Show send and receive errors
--verbose, -v	Increase verbosity level
--debug, -d	Paise debug level
--help, -h	Help message

Table 41 - perfquery Options

Flag	Description
--version, -V	Show version

Examples

```
perfquery -r 32 1       # read performance counters and reset
perfquery -e -r 32 1    # read extended performance counters and reset
perfquery -R 0x20 1     # reset performance counters of port 1 only
perfquery -e -R 0x20 1  # reset extended performance counters of port 1 only
perfquery -R -a 32      # reset performance counters of all ports
perfquery -R 32 2 0x0fff        # reset only error counters of port 2
perfquery -R 32 2 0xf000        # reset only non-error counters of port 2
```

1. Read local port's performance counters.

```
> perfquery
# Port counters: Lid 6 port 1
PortSelect:......................1
CounterSelect:...................0x1000
SymbolErrors:....................0
LinkRecovers:....................0
LinkDowned:......................0
RcvErrors:.......................0
RcvRemotePhysErrors:.............0
RcvSwRelayErrors:................0
XmtDiscards:.....................0
XmtConstraintErrors:.............0
RcvConstraintErrors:.............0
LinkIntegrityErrors:.............0
ExcBufOverrunErrors:.............0
VL15Dropped:.....................0
XmtData:.........................55178210
RcvData:.........................55174680
XmtPkts:.........................766366
RcvPkts:.........................766315
```

2. Read performance counters from LID 2, all ports.

```
> smpquery -a 2
# Port counters: Lid 2 port 255
PortSelect:......................255
CounterSelect:...................0x0100
SymbolErrors:....................65535
LinkRecovers:....................255
LinkDowned:......................16
RcvErrors:.......................657
RcvRemotePhysErrors:.............0
```

```
RcvSwRelayErrors:.............70
XmtDiscards:..................488
XmtConstraintErrors:..........0
RcvConstraintErrors:..........0
LinkIntegrityErrors:..........0
ExcBufOverrunErrors:..........0
VL15Dropped:..................0
XmtData:......................129840354
RcvData:......................129529906
XmtPkts:......................1803332
RcvPkts:......................1799018
```

3. Read then reset performance counters from LID 2, port 1.

```
> perfquery  -r 2 1
# Port counters: Lid 2 port 1
PortSelect:...................1
CounterSelect:................0x0100
SymbolErrors:.................0
LinkRecovers:.................0
LinkDowned:...................0
RcvErrors:....................0
RcvRemotePhysErrors:..........0
RcvSwRelayErrors:.............0
XmtDiscards:..................3
XmtConstraintErrors:..........0
RcvConstraintErrors:..........0
LinkIntegrityErrors:..........0
ExcBufOverrunErrors:..........0
VL15Dropped:..................0
XmtData:......................0
RcvData:......................0
XmtPkts:......................0
RcvPkts:......................0
```

9.4.18 mstflint

Queries and burns a binary firmware-image file on non-volatile (Flash) memories of Mellanox InfiniBand and Ethernet network adapters. The tool requires root privileges for Flash access.

If you purchased a standard Mellanox Technologies network adapter card, please download the firmware image from www.mellanox.com > Downloads > Firmware. If you purchased a non-standard card from a vendor other than Mellanox Technologies, please contact your vendor.

To run mstflint, you must know the device location on the PCI bus. See Example 1 for details.

Synopsis

```
mstflint [switches...] <command> [parameters...]
```

Output Files

Table 42 lists the various switches of the utility, and Table 43 lists its commands.

Table 42 - mstflint Switches (Sheet 1 of 2)

Switch	Affected/ Relevant Commands	Description
-h		Print the help menu
-hh		Print an extended help menu
-d[evice] <device>	All	Specify the device to which the Flash is connected.
-guid <GUID>	burn, sg	GUID base value. 4 GUIDs are automatically assigned to the following values: guid -> node GUID guid+1 -> port1 guid+2 -> port2 guid+3 -> system image GUID. Note: Port2 guid will be assigned even for a single port HCA; the HCA ignores this value.
-guids <GUIDs...>	burn, sg	4 GUIDs must be specified here. The specified GUIDs are assigned the following values, repectively: node, port1, port2 and system image GUID. Note: Port2 guid must be specified even for a single port HCA; the HCA ignores this value. It can be set to 0x0.
-mac <MAC>	burn, sg	MAC address base value. Two MACs are automatically assigned to the following values: mac -> port1 mac+1 -> port2 Note: This switch is applicable only for Mellanox Technologies Ethernet products.
-macs <MACs...>	burn, sg	Two MACs must be specified here. The specified MACs are assigned to port1 and port2, repectively. Note: This switch is applicable only for Mellanox Technologies Ethernet products.
-blank_guids	burn	Burn the image with blank GUIDs and MACs (where applicable). These values can be set later using the sg command – see Table 43 below.
-clear_semaphore	No commands allowed	Force clear the Flash semaphore on the device. No command is allowed when this switch is used. Warning: May result in system instability or Flash corruption if the device or another application is currently using the Flash.
-i[mage] <image>	burn, verify	Binary image file

Table 42 - mstflint Switches (Sheet 2 of 2)

Switch	Affected/ Relevant Commands	Description
-qq	burn, query	Run a quick query. When specified, mstflint will not perform full image integrity checks during the query operation. This may shorten execution time when running over slow interfaces (e.g., I2C, MTUSB-1).
-nofs	burn	Burn image in a non-failsafe manner
-skip_is	burn	Allow burning the firmware image without updating the invariant sector. This is to ensure failsafe burning even when an invariant sector difference is detected.
-byte_mode	burn, write	Shift address when accessing Flash internal registers. May be required for burn/write commands when accessing certain Flash types.
-s[ilent]	burn	Do not print burn progress messages
-y[es]	All	Non-interactive mode: Assume the answer is “yes” to all questions.
-no	All	Non-interactive mode: Assume the answer is “no” to all questions.
-vsd <string>	burn	Write this string of up to 208 characters to VSD upon a burn command.
-use_image_ps	burn	Burn vsd as it appears in the given image - do not keep existing VSD on Flash.
-dual_image	burn	Make the burn process burn two images on Flash. The current default failsafe burn process burns a single image (in alternating locations).
-v		Print version info

Table 43 - mstflint Commands (Sheet 1 of 2)

Command	Description
b[urn]	Burn Flash
q[uery]	Query miscellaneous Flash/firmware characteristics
v[erify]	Verify the entire Flash
bb	Burn Block: Burn the given image as is, without running any checks

Table 43 - mstflint Commands (Sheet 2 of 2)

Command	Description
sg	Set GUIDs
ri <out-file>	Read the firmware image on the Flash into the specified file
dc <out-file>	Dump Configuration: Print a firmware configuration file for the given image to the specified output file
e[rase] <addr>	Erase sector
rw <addr>	Read one DWORD from Flash
ww <addr> < data>	Write one DWORD to Flash
wwne <addr>	Write one DWORD to Flash without sector erase
wbne <addr> <size> <data...>	Write a data block to Flash without sector erase
rb <addr> <size> [out-file]	Read a data block from Flash
swreset	SW reset the target InfniScale® IV device. This command is supported only in the In-Band access method.

Possible command return values are:

```
0 - successful completion
1 - error has occurred
7 - the burn command was aborted because firmware is current
```

Examples

1. Find Mellanox Technologies's ConnectX® VPI cards with PCI Express running at 2.5GT/s and InfiniBand ports at DDR / or Ethernet ports at 10GigE.

```
> /sbin/lspci -d 15b3:634a
04:00.0 InfiniBand: Mellanox Technologies MT25418 [ConnectX IB DDR, PCIe 2.0 2.5GT/s]
(rev a0).
```

   In the example above, 15b3 is Mellanox Technologies's vendor number (in hexadecimal), and 634a is the device's PCI Device ID (in hexadecimal). The number string 04:00.0 identifies the device in the form bus:dev.fn.

The PCI Device IDs of Mellanox Technologies' devices can be obtained from the PCI ID Repository Website at http://pci-ids.ucw.cz/read/PC/15b3.

2. Verify the ConnectX firmware using its ID (using the results of the example above).

```
> mstflint -d 04:00.0 v
     ConnectX failsafe image. Start address: 80000. Chunk size 80000:

NOTE: The addresses below are contiguous logical addresses. Physical addresses on flash
may be different, based on the image start address and chunk size
```

```
     /0x00000038-0x000010db (0x0010a4)/ (BOOT2) - OK
     /0x000010dc-0x00004947 (0x00386c)/ (BOOT2) - OK
     /0x00004948-0x000052c7 (0x000980)/ (Configuration) - OK
     /0x000052c8-0x0000530b (0x000044)/ (GUID) - OK
     /0x0000530c-0x0000542f (0x000124)/ (Image Info) - OK
     /0x00005430-0x0000634f (0x000f20)/ (DDR) - OK
     /0x00006350-0x0000f29b (0x008f4c)/ (DDR) - OK
     /0x0000f29c-0x0004749b (0x038200)/ (DDR) - OK
     /0x0004749c-0x0005913f (0x011ca4)/ (DDR) - OK
     /0x00059140-0x0007a123 (0x020fe4)/ (DDR) - OK
     /0x0007a124-0x0007bdff (0x001cdc)/ (DDR) - OK
     /0x0007be00-0x0007eb97 (0x002d98)/ (DDR) - OK
     /0x0007eb98-0x0007f0af (0x000518)/ (Configuration) - OK
     /0x0007f0b0-0x0007f0fb (0x00004c)/ (Jump addresses) - OK
     /0x0007f0fc-0x0007f2a7 (0x0001ac)/ (FW Configuration) - OK

FW image verification succeeded. Image is bootable.
```

9.4.19 ibv_asyncwatch

Display asynchronous events forwarded to userspace for an InfiniBand device.

Synopsis

```
ibv_asyncwatch
```

Examples

1. Display asynchronous events.

```
> ibv_asyncwatch
mlx4_0: async event FD 4
```

9.4.20 ibdump

Dump InfiniBand traffic that flows to and from Mellanox Technologies ConnectX® family adapters InfiniBand ports. The dump file can be loaded by the Wireshark tool for graphical traffic analysis.

The following describes a work flow for local HCA (adapter) sniffing:

- Run ibdump with the desired options
- Run the application that you wish its traffic to be analyzed
- Stop ibdump (CTRL-C) or wait for the data buffer to fill (in --mem-mode)
- Open Wireshark and load the generated file

How to Get Wireshark:

Download the current release from www.wireshark.org for a Linux or Windows environment. See the ibdump_release_notes.txt file for more details.

Although ibdump is a Linux application, the generated .pcap file may be analyzed on either operating system.

In order for ibdump to function with RoCE, Flow Steering must be enabled.
To do so, add the following to `/etc/modprobe.d/mlnx.conf` file:
`options mlx4_core log_num_mgm_entry_size=-1` and then restart the drivers.

Synopsis

```
ibdump [options]
```

Output Files

```
-d, --ib-dev=<dev>            use RDMA device <dev> (default first device found)
                              The relevant devices can be listed by running the
                              'ibv_devinfo' command.
-i, --ib-port=<port>          use port <port> of IB device (default 1)
```

```
-w, --write=<file>              dump file name (default "sniffer.pcap")
                                '-' stands for stdout - enables piping to tcpdump or
                                tshark.
-o, --output=<file>             alias for the '-w' option. Do not use - for backward
                                compatibility
-b, --max-burst=<log2 burst>    og2 of the maximal burst size that can be captured with
l                               no packets loss.
                                Each entry takes ~ MTU bytes of memory (default 12 -
                                4096 entries)
-s, --silent                    do not print progress indication.
--mem-mode <size>               when specified, packets are written to file only after
                                the capture is stopped. It is faster than default mode
                                (less chance for packet loss), but takes more memory. In
                                this mode, ibdump stops after <size> bytes are captured
--decap                         Decapsulate port mirroring headers. Should be used when
                                capturing RSPAN traffic.
-h, --help                      Display this help screen.
-v, --version                   Print version information.
```

Examples

- Run ibdump.

```
> ibdump
 ------------------------------------------------
 IB device                     : "mlx4_0"
 IB port                       : 1
 Dump file                     : sniffer.pcap
 Sniffer WQEs (max burst size)  : 4096
 ------------------------------------------------

Initiating resources ...
searching for IB devices in host
Port active_mtu=2048
MR was registered with addr=0x60d850, lkey=0x28042601, rkey=0x28042601, flags=0x1
QP was created, QP number=0x4004a

Ready to capture (Press ^c to stop):
```

9.4.21 ibnodes

Uses the current InfiniBand subnet topology or an already saved topology file and extracts the InfiniBand nodes (CAs and switches).

Synopsis

```
ibnodes [<topology-file>]
```

9.4.22 ibswitches

Traces the InfiniBand subnet topology or uses an already saved topology file to extract the InfiniBand switches.

Synopsis

```
ibswitches [-h] [<topology-file>]
```

9.4.23 ibnetsplit

Automatically groups hosts and creates scripts that can be run in order to split the network into sub-networks containing one group of hosts.

Synopsis

- Group

```
ibnetsplit [-v][-h][-g grp-file] -s <.lst|.net|.topo> <-r head-ports|-d max-dist>
```

- Split

```
ibnetsplit [-v][-h][-g grp-file] -s <.lst|.net|.topo> -o out-dir
```

- Combined

```
ibnetsplit [-v][-h][-g grp-file] -s <.lst|.net|.topo> <-r head-ports|-d max-dist> -o
out-dir
```

Usage

- Grouping:

  The grouping is performed if the -r or -d options are provided.

  - If the -r is provided with a file containing group head ports, the algorithm examines the hosts distance from the set of node ports provided in the head-ports file (these are expected to be the ports running standby SM's).
  - If the -d is provided with a maximum distance of the hosts in each group, the algorithm partition the hosts by that distance.

This method of analyzation may not be suitable for some topologies.

  The results of the identified groups are printed into the file defined by the -g option (default ibnetsplit.groups) and can be manually edited. For groups where the head port is a switch, the group file uses the FIRST host port as the port to run the isolation script from.

- Splitting:

  If the -o flag is included, this algorithm analyzes the MinHop table of the topology and identifies the set of links and switches that may potentially be used for routing each group ports. The cross-switch links between switches of the group to other switches are declared as split-links and the commands to turn them off using Directed Routes from the original Group Head ports are written into the out-dir provided by the -o flag.

Both stages require a subnet definition file to be provided by the -s flag. The supported formats for subnet definition are:

- *.net - for ibnetdiscover
- *.lst - for opensm-subnet.lst or ibiagnet.lst
- *.topo - for a topology file

HEAD PORTS FILE

This file is provided by the user and defines the ports by which grouping of the other host ports is defined.

Format:

Each line should contain either the name or the GUID of a single port. For switches the port number shall be 0.

```
<node-name>/P<port-num>|<PGUID>
```

GROUPS FILE

This file is generated by the program if the head-ports file is provided to it. Alternatively it can be provided (or edited) by the user if different grouping is desired. The generated script for isolating or connecting the group should be run from the first node in each group.

Format:

Each line may be either:

```
GROUP: <group name>
<node-name>/P<port-num>|<PGUID>
```

9.4.24 ibccconfig

Supports the configuration of congestion control settings on switches and HCAs.

Synopsis

```
ibccconfig [options] [-c key] <op> <lid|guid>
```

Operations

Operation	Parameters
CongestionKeyInfo (CK)	<lid|guid> <cckey> <cckeyprotectbit> <cckeyleaseperiod> <cckeyviolations>
SwitchCongestionSetting (SS)	<lid|guid> <controlmap> <victimmask> <creditmask> <threshold> <packetsize> <csthreshold> <csreturndelay> <markingrate>
SwitchPortCongestionSetting (SP)	<lid|guid> <portnum> <valid> <control_type> <threshold> <packet_size> <cong_parm_marking_rate>
CACongestionSetting (CS)	<lid|guid> <port_control> <control_map> <ccti_timer> <ccti_increase> <trigger_threshold> <ccti_min>

Operation	Parameters
CongestionControlTable (CT)	<lid|guid> <cctilimit> <index> <cctentry> <cctentry>

Options

```
--cckey, -c <key>        CC key
--config, -z <config>    use config file, default: /etc/infiniband-diags/ibdiag.conf
--Ca, -C <ca>            Ca name to use
--Port, -P <port>        Ca port number to use
--Lid, -L                use LID address argument
--Guid, -G               use GUID address argument
--timeout, -t <ms>       timeout in ms
--sm_port, -s <lid>      SM port lid
--m_key, -y <key>        M_Key to use in request
--errors, -e             show send and receive errors
--verbose, -v            increase verbosity level
--debug, -d              raise debug level
--help, -h               help message
--version, -V            show version
```

Examples

```
ibccconfig SwitchCongestionSetting 2 0x1F          # Configure Switch Congestion Settings
0x1FFFFFFFFF 0x0 0xF 8 0 0:0 1
ibccconfig CACongestionSetting 1 0 0x3 150 1 0 0   # Configure CA Congestion Settings to SL 0 and
                                                   SL 1
ibccconfig CACongestionSetting 1 0 0x4 200 1 0 0   # Configure CA Congestion Settings to SL 2
ibccconfig CongestionControlTable 1 63 0 0:0 0:1   # Configure first block of Congestion Control
...                                                Table
ibccconfig CongestionControlTable 1 127 0 0:64     # Configure second block of Congestion Control
0:65 ...                                           Table
```

9.4.25 ibccquery

Supports the querying of settings and other information related to congestion control.

Synopsis

```
ibccquery [options] [-c key] <op> <lid|guid>
```

Operations

Operation	Parameters
ClassPortInfo (CP)	<lid|guid>
CongestionInfo (CI)	<lid|guid>
CongestionKeyInfo (CK	<lid|guid>

Operation	Parameters
CongestionLog (CL)	<lid|guid>
SwitchCongestionSetting (SS)	<lid|guid>
SwitchPortCongestionSetting (SP)	<lid|guid> [<portnum>]
CACongestionSetting (CS)	<lid|guid>
CongestionControlTable (CT)	<lid|guid>
Timestamp (TI)	<lid|guid>

Options

```
--cckey, -c <key>       CC key
--config, -z <config>   use config file, default: /etc/infiniband-diags/ibdiag.conf
--Ca, -C <ca>           Ca name to use
--Port, -P <port>       Ca port number to use
--Lid, -L               use LID address argument
--Guid, -G              use GUID address argument
--timeout, -t <ms>      timeout in ms
--sm_port, -s <lid>     SM port lid
--m_key, -y <key>       M_Key to use in request
--errors, -e            show send and receive errors
--verbose, -v           increase verbosity level
--debug, -d             raise debug level
--help, -h              help message
--version, -V           show version
```

Examples

```
ibccquery CongestionInfo 3                  # Congestion Info by lid
ibccquery SwitchPortCongestionSetting 3     # Query all Switch Port Congestion Settings
ibccquery SwitchPortCongestionSetting 3 1   # Query Switch Port Congestion Setting for port 1
```

9.4.26 smparquery

The tool sends smp query for adaptive routing and private LFT features.

Synopsis

```
smparquery <op> <dest dr_path|lid> [op params]
```

Operations

Operation	Parameters
ARInfo (ARI)	<addr>

Operation	Parameters
ARGroupTable (ARGT)	<addr> [<plft>] [<group_table>] [<blocknum>]
ARLFTTable (ARLT)	<addr> [<plft>] [<blocknum>]
PLFTInfo (PLFTI)	<addr>
PLFTDef (PLFTD)	<addr> [<blocknum>]
PortSLToPLFTMap (PLFTP)	<addr> [<blocknum>]

Options

```
-C|--Ca <ca>        Ca name to use
-P|--Port <port>   Ca port number to use
-D|--Direct         use Direct address argument
-L|--Lid           use LID address argument
-h|--help          help message
-V|--version       show version
```

Examples

```
smparquery argrouptable 3 1                # argrouptable by lid, with block modifier
smparquery -D arinfo 0,1,2,3               # arinfo by direct route
```

9.4.27 ibtopodiff

The tool compares a topology file and a discovered listing ofsubnet.lst/ibdiagnet.lst and reports missmatches.

Two different algorithms provided:

- Using the -e option is more suitible for MANY mismatches it applies less heuristics and provide details about the match.
- Providing the -s, -p and -g starts a detailed heuristics that should be used when only small number of changes are expected.

Synopsis

```
ibtopodiff [-v][-h][-m map-file][-c csv-file] -t <topo file> -d <subnet file> {-e | -s <start
node name> -p <start port num> -g <start port guid>}
```

Options

```
-c|--csv out-file   optionally output into the given CSV file.
-m|--map map-file   mapping topo node to discover node name/GUID
-v|--verbose        verbsoe mode
-h|--help           provide this help message
```

Additional Options

```
-t|--topology <file>      Topology file [ibdm.topo]. The format is defined in the IBDM user
                          manual.
-d|--discovered <file>    [subnet.lst] file produced by OpenSM.
-e|--edge                 start processing from the edges using strict match.
-s|--start-node <name>    The name of the start node [H-1/U1].
-p|--port-num <num>       The number of the start port [1].
-g|--port-guid <guid>     The guid of the start port [none].
```

9.4.28 ibcongest

Provides static congestion analysis. It calculates routing for a given topology (topo-mode) or uses extracted lst/fdb files (lst-mode). Additionally, it analyzes congestion for a traffic schedule provided in a "schedule-file" or uses an automatically generated schedule of all-to-all-shift.

To display a help message which details the tool's options, please run "/opt/ibutils2/bin/ibcongest -h".

Synopsis

TOPO

```
ibcongest [-hdABC][-v lvl] -t <topo-file> [-n <sm-node>][-p <sm-port>] [-r ftree|pgft|osm>]
[-z roots-file] [-R rep-hsd-file] [-Q rep-bw-file] [-O order-file] [-o order-file] {-a|-s
<schedule>}
```

LST

```
ibcongest [-hdABC][-v lvl] -l <lst> -f <fdbs> [-R rep-hsd-file] [-Q rep-bw-file]
[-O order-file] [-o order-file] {-a [-j <min-shift> -k <max-shift>]|-s <schedule>}
```

TOPO Mode Options

```
-t topo-file          [Mandatory] Topology file to be used.
-r ftree/pgft/osm     Defines the routing engine to be used (osm).
-n sm-node            Defines the node name for the SM to run on (H-1/U1).
-p sm-port            Defines the port number of the sm-node the SM run from (1).
```

LST Mode Options

```
-l lst-file           [Mandatory] LST file obtained from OpenSM or ibdiagnet.
-f fdbs-file          [Mandatory] FDBs file obtained from OpenSM or ibdiagnet.
```

Options

```
-A                      Checks all paths not just CA to CA
-B                      Calculates and reports actual bandwidth of each flow. Turned on also
                        if -Q is provided.
-C                      Assumes Congestion Control. I.e. flow fairness and no back-pressure.
-F                      Estimates flow fractions in HSD report. NOTE: this is inaccurate as
                        it does not take effects of HoL blocking or round-robin arbitration
                        into account
```

-O order-file	Runa a simulated annealing and write out the result into the given file.
-o order-file	A file holding host name in each line, defining their ordering. Description of the order file is in section ORDER_FILE. Default order is by host name sort H-<N> by the N.
-m	Ignores missing CA-to-CA paths
-R rep-hsd-file	Report each stage over subscription into the given file name
-Q rep-bw-file	Reports each stage each flow BW (in csb format) into the given file name
-v lvl: verbose mode	Prepares for being flooded... lvl is: 1 - 2 - order optimization 3 - congestion analysis 4 - fat tree routing 5 - fat tree recognition
-h	This message

Additional Options

-s schedule-file	A file defining the communication pattern. The format described in the below section SCHEDULE FILE.
-a	Automatically generate an all-to-all schedule
-j min-shift -k max-shift	Automatically generate a range of shifts
-z roots-file	A file holding names of switches which are the roots of the topology. This is mandatory if the -F flag is provided to report HSD while estimating flow fractions.

SCHEDULE FILE

The schedule file describes a communication sequence. There are 2 types of lines:

- Stage line separates different stages of the schedule:

```
^STAGE(\s+[0-9]+)?
```

- SRC DST line describes traffic in pairs. Optionally the destination may be given a specific 'path-bits' that specifies the exact DLID from a range of DLIDs a destination may have. Another option is the flow may be given a relative BW of the SRC node.
  - From the first rank to the second rank:

```
^\s*[0-9]+\s+[0-9]+(:([0-9]+))?(\s+(0\.[0-9]+))?\s*
```

  - From a port name to a port name:

```
^\s*(\S+)/P([0-9]+)\s+*(\S+)/P([0-9]+)(:([0-9]+))?(\s+(0\.[0-9]+))?\s*
```

    For example:

    A flow from node H-1 port 1 to H-100 port 1 that 50% of the BW of H-1

```
H-1/U1/P1 H-100/U1/P1:1 0.5
```

ORDER FILE

The order file provides mapping between the rank (which is the line number) and a host in the topology. The following are the type of lines supported:

- OpenSM order file style holding LID and host name

```
^\s*(0x[0-9a-fA-F]+)\s(\s+)\s+HCA-([0-9]+)
```

- Port name only: simply declare the order of host names to be used

```
^\s*(\s+)\s+P?([0-9]+)\s*$
```

- Dummy place holder

```
^DUMMY$
```

DUMMY stands for an empty rank. Just a placeholder - not a real HCA. Traffic from DUMMY or to DUMMY is not generated even if specified in the schedule.

9.4.29 ibaddr

Can be used to show the LID and GID addresses of the specified port or the local port by default. This utility can be used as simple address resolver.

Syntax

```
ibaddr [options] [<dest_addr>]
```

Non standard flags:

```
gid_show  (-g):  Show gid address only
lid_show   (-l): Show lid range only
Lid_show (-L):   Show lid range (in decimal) only
```

Examples:

```
ibaddr                # show local address
ibaddr 2               # show address of the specified port lid
ibaddr -G 0x8f1040023  # show address of the specified port guid
```

9.4.30 sminfo

Issues and dumps the output of an sminfo query in human readable format. The target SM is the one listed in the local port info or the SM specified by the optional SM LID or by the SM direct routed path.

Using sminfo for any purpose other than a simple query might result in a malfunction of the target SM.

Syntax

```
sminfo [options] <sm_lid|sm_dr_path> [sminfo_modifier]
```

Non standard flags

```
-s <state>          # use the specified state in sminfo mad
-p <priority>       # use the specified priority in sminfo mad
-a <activity>       # use the specified activity in sminfo mad
```

Examples:

```
sminfo         # show sminfo of SM listed in local portinfo
sminfo 2       # query SM on port lid 2
```

9.4.31 ibnetdiscover

Performs InfiniBand subnet discovery and outputs a human readable topology file. GUIDs, node types, and port numbers are displayed as well as port LIDs and node descriptions. All nodes (and links) are displayed (full topology). This utility can also be used to list the current connected nodes. The output is printed to the standard output unless a topology file is specified.

Syntax

```
ibnetdiscover [options] [<topology-filename>]
```

Non standard flags:

```
l                       Lists connected nodes
H                       Lists connected HCAs
S                       Lists connected switches
g                       Groups
full (-f)               Shows full information (ports' speed and width, vlcap)
show (-s)               Shows more information
Router_list  (-R)       Lists connected routers
node-name-map           Nodes name map file
cache                   filename to cache ibnetdiscover data to
load-cache              filename of ibnetdiscover cache to load
diff                    filename of ibnetdiscover cache to diff
diffcheck               Specifies checks to execute for --diff
ports : (-p)            Obtains a ports report
max_hops (-m)           Reports max hops discovered by the library
outstanding_smps (-o)   Specifies the number of outstanding SMP's which should be issued
                        during the scan
```

9.4.32 ibping

Uses vendor mads to validate connectivity between InfiniBand nodes. On exit, (IP) ping like output is show. ibping is run as client/server. The default is to run as client. Note also that a default ping server is implemented within the kernel.

Syntax

```
ibping [options] <dest lid|guid>
```

Non standard flags:

```
-c <count>      stop after count packets
-f              flood destination: send packets back to back w/o delay
-o <oui>        use specified OUI number to multiplex vendor MADs
-S              start in server mode (do not return)
```

9.4.33 ibsysstat

Uses vendor mads to validate connectivity between InfiniBand nodes and obtain other information about the InfiniBand node. ibsysstat is run as client/server. The default is to run as client.

Syntax

```
ibsysstat [options] <dest lid|guid> [<op>]
```

Non standard flags:

```
ping     Verify connectivity to server (default)
host     Obtain host information from server
cpu      Obtain cpu information from server
-o <oui>    Use specified OUI number to multiplex vendor mads
-S          Start in server mode (do not return)
```

9.4.34 ibcacheedit

ibcacheedit allows users to edit an ibnetdiscover cache created through the --cache option in ibnetdiscover(8).

Syntax

```
ibcacheedit [options] <orig.cache> <new.cache>
```

OPTIONS

--switchguid BEFOREGUID:AFTERGUID	Specify a switchguid that should be changed. The before and after guid should be separated by a colon. On switches, port guids are identical to the switch guid, so port guids will be adjusted as well on switches.
--caguid BEFOREGUID:AFTERGUID	Specify a caguid that should be changed. The before and after guid should be separated by a colon.
--sysimgguid BEFOREGUID:AFTERGUID	Specify a sysimgguid that should be changed. The before and after guid should be spearated by a colon.
--portguid NODEGUID:BEFORE-GUID:AFTERGUID	Specify a portguid that should be changed. The nodeguid of the port (e.g. switchguid or caguid) should be specified first, followed by a colon, the before port guid, another colon, then the after port guid. On switches, port guids are identical to the switch guid, so the switch guid will be adjusted as well on switches.

Debugging flags

```
-h, --help        show the usage message
-V, --version     show the version info.
```

9.4.35 dump_fts

Dumps tables for every switch found in an ibnetdiscover scan of the subnet. The dump file format is compatible with loading into OpenSM using the -R file -U /path/to/dump-file syntax.

Syntax

```
dump_fts [options] [<startlid> [<endlid>]]
```

Options

```
-a, --all           show all lids in range, even invalid entries
-n, --no_dests      do not try to resolve destinations
-M, --Multicast     show multicast forwarding tables In this case, the range parameters are
                    specifying the mlid range.
```

Port Selection flags

```
-C, --Ca <ca_name>     use the specified ca_name.
-P, --Port <ca_port>   use the specified ca_port.
```

Local port Selection

Multiple port/Multiple CA support: when no IB device or port is specified (see the "local umad parameters" below), the libibumad library selects the port to use by the following criteria:

- The first port that is ACTIVE.
- If not found, the first port that is UP (physical link up).
- If a port and/or CA name is specified, the libibumad library attempts to fulfill the user request, and will fail if it is not possible.

For example:

```
ibaddr            # use the first port (criteria #1 above)
ibaddr -C mthca1  # pick the best port from "mthca1" only.
ibaddr -P 2       # use the second (active/up) port from the first available IB device.
ibaddr -C mthca0 -P 2  # use the specified port only.
```

Debugging flags

```
-d              raise the IB debugging level.  May be used several times (-ddd or -d -d -d).
-e              show send and receive errors (timeouts and others)
-h, --help      show the usage message
-v, --verbose   increase the application verbosity level.  May be used several times (-vv or
                -v -v -v)
-V, --version   show the version info.
```

Configuration flags

```
-t, --timeout <timeout_ms>       override the default timeout for the solicited mads.
-y, --m_key <key>                use the specified M_key for requests. If non-numeric value
                                 (like 'x') is specified then a value will be prompted for.
```

```
--node-name-map <node-name-map>   Specify a node name map. This file maps GUIDs to more user
                                  friendly names.  See FILES section.
--config, -z  <config_file>       Specify alternate config file. Default: /etc/infiniband-
                                  diags/ibdiag.conf
```

CONFIG FILE

```
/etc/infiniband-diags/ibdiag.conf
```

A global config file is provided to set some of the common options for all tools. See supplied config file for details.

NODE NAME MAP FILE FORMAT

The node name map is used to specify user friendly names for nodes in the output. GUIDs are used to perform the lookup.

This functionality is provided by the opensm-libs package. See opensm(8) for the file location for your installation.

Generically:

```
# comment
<guid> "<name>"
```

Examples:

```
# IB1
# Line cards
0x0008f104003f125c "IB1 (Rack 11 slot 1   ) ISR9288/ISR9096 Voltaire sLB-24D"
0x0008f104003f125d "IB1 (Rack 11 slot 1   ) ISR9288/ISR9096 Voltaire sLB-24D"
0x0008f104003f10d2 "IB1 (Rack 11 slot 2   ) ISR9288/ISR9096 Voltaire sLB-24D"
0x0008f104003f10d3 "IB1 (Rack 11 slot 2   ) ISR9288/ISR9096 Voltaire sLB-24D"
0x0008f104003f10bf "IB1 (Rack 11 slot 12  ) ISR9288/ISR9096 Voltaire sLB-24D"

# Spines
0x0008f10400400e2d "IB1 (Rack 11 spine 1   ) ISR9288 Voltaire sFB-12D"
0x0008f10400400e2e "IB1 (Rack 11 spine 1   ) ISR9288 Voltaire sFB-12D"
0x0008f10400400e2f "IB1 (Rack 11 spine 1   ) ISR9288 Voltaire sFB-12D"
0x0008f10400400e31 "IB1 (Rack 11 spine 2   ) ISR9288 Voltaire sFB-12D"
0x0008f10400400e32 "IB1 (Rack 11 spine 2   ) ISR9288 Voltaire sFB-12D"

# GUID   Node Name
0x0008f10400411a08 "SW1  (Rack  3) ISR9024 Voltaire 9024D"
0x0008f10400411a28 "SW2  (Rack  3) ISR9024 Voltaire 9024D"
0x0008f10400411a34 "SW3  (Rack  3) ISR9024 Voltaire 9024D"
0x0008f104004119d0 "SW4  (Rack  3) ISR9024 Voltaire 9024D"
```

9.5 Performance Utilities

The performance utilities described in this chapter are intended to be used as a performance micro-benchmark.

9.5.1 ib_read_bw

ib_read_bw calculates the BW of RDMA read between a pair of machines. One acts as a server and the other as a client. The client RDMA reads the server memory and calculate the BW by sampling the CPU each time it receive a successful completion. The test supports features such as Bidirectional, in which they both RDMA read from each other memory's at the same time, change of mtu size, tx size, number of iteration, message size and more. Read is available only in RC connection mode (as specified in IB spec).

Synopsis

Server

```
ib_read_bw [options]
```

Client

```
ib_read_bw [options] <hostname>
```

Options

The table below lists the various flags of the command.

Table 44 - ib_read_bw Flags and Options

Flag	Description
-a, --all	Run sizes from 2 till 2^23
-b, --bidirectional	Measure bidirectional bandwidth (default unidirectional)
-c, --connection=<RC/XRC/DC>	Connection type RC/XRC/DC (default RC)
-d, --ib-dev=<dev>	Use IB device <dev> (default first device found)
-D, --duration	Run test for a customized period of seconds.
-e, --events	Sleep on CQ events (default poll)
-f, --margin	Measure results within margins. (default=2sec)
-F, --CPU-freq	Do not fail even if cpufreq_ondemand module is loaded
-h, --help	Show this help screen.
-i, --ib-port=<port>	Use port <port> of IB device (default 1)
-l, --post_list=<list size>	Post list of WQEs of <list size> size (instead of single post)
-m, --mtu=<mtu>	MTU size : 256 - 4096 (default port mtu)
-n, --iters=<iters>	Number of exchanges (at least 5, default 1000)

Table 44 - ib_read_bw Flags and Options

Flag	Description
-N, --no peak-bw	Cancel peak-bw calculation (default with peak)
-o, --outs=<num>	Number of outstanding read/atom(default max of device)
-O, --dualport	Run test in dual-port mode.
-p, --port=<port>	Listen on/connect to port <port> (default 18515)
-q, --qp=<num of qp's>	Number of qp's (default 1)
-Q, --cq-mod	Generate Cqe only after <--cq-mod> completion
-R, --rdma_cm	Connect QPs with rdma_cm and run test on those QPs
-s, --size=<size>	Size of message to exchange (default 65536)
-S, --sl=<sl>	SL (default 0)
-t, --tx-depth=<dep>	Size of tx queue (default 128)
-T, --tos=<tos value>	Set <tos_value> to RDMA-CM QPs. available only with -R flag. values 0-256 (default off)
-u, --qp-timeout=<timeout>	QP timeout, timeout value is 4 usec * 2 ^(timeout), default 14
-V, --version	Display version number
-w, --limit_bw	Set verifier limit for bandwidth
-x, --gid-index=<index>	Test uses GID with GID index (Default : IB - no gid . ETH - 0)
-y, --limit_msgrate	Set verifier limit for Msg Rate
-z, --com_rdma_cm	Communicate with rdma_cm module to exchange data - use regular QPs

Additional Options

The table below lists the additional flags of the command.

Table 45 - Additional ib_read_bw Flags and Options

Flag	Description
--inline_recv=<size>	Max size of message to be sent in inline receive
--output=<units>	Set verbosity output level: bandwidth , message_rate, latency_typical
--pkey_index=<pkey index>	PKey index to use for QP

Table 45 - Additional ib_read_bw Flags and Options

Flag	Description
--report-both	Report RX & TX results separately on Bidirectinal BW tests
--report_gbits	Report Max/Average BW of test in Gbit/sec (instead of MB/sec)
--run_infinitely	Run test forever, print results every <duration> seconds

9.5.2 ib_read_lat

ib_read_lat calculates the latency of RDMA read operation of message_size between a pair of machines. One acts as a server and the other as a client. They perform a ping pong benchmark on which one side RDMA reads the memory of the other side only after the other side have read his memory. Each of the sides samples the CPU clock each time they read the other side memory , in order to calculate latency. Read is available only in RC connection mode (as specified in IB spec).

Synopsis

Server

```
ib_read_lat [options]
```

Client

```
ib_read_lat [options] <hostname>
```

Options

The table below lists the various flags of the command.

Table 46 - ib_read_lat Flags and Options

Flag	Description
-a, --all	Run sizes from 2 till 2^23
-c, --connection=<RC/XRC/DC>	Connection type RC/XRC/DC (default RC)
-C, --report-cycles	Report times in cpu cycle units (default microseconds)
-d, --ib-dev=<dev>	Use IB device <dev> (default first device found)
-D, --duration	Run test for a customized period of seconds.
-e, --events	Sleep on CQ events (default poll)
-f, --margin	Measure results within margins. (default=2sec)
-h, --help	Show this help screen.
-H, --report-histogram	Print out all results (default print summary only)
-i, --ib-port=<port>	Use port <port> of IB device (default 1)

Table 46 - ib_read_lat Flags and Options

Flag	Description
-m, --mtu=<mtu>	MTU size : 256 - 4096 (default port mtu)
-n, --iters=<iters>	Number of exchanges (at least 5, default 1000)
-o, --outs=<num>	Num of outstanding read/atom (default max of device)
-p, --port=<port>	Listen on/connect to port <port> (default 18515)
-R, --rdma_cm	Connect QPs with rdma_cm and run test on those QPs
-s, --size=<size>	Size of message to exchange (default 2)
-S, --sl=<sl>	SL (default 0)
-T, --tos=<tos value>	Set <tos_value> to RDMA-CM QPs. availible only with -R flag. values 0-256 (default off)
-u, --qp-timeout=<timeout>	QP timeout, timeout value is 4 usec * 2 ^(timeout), default 14
-U, --report-unsorted	(implies -H) print out unsorted results (default sorted)
-V, --version	Display version number
-x, --gid-index=<index>	Test uses GID with GID index (Default : IB - no gid . ETH - 0)
-z, --com_rdma_cm	Communicate with rdma_cm module to exchange data - use regular QPs

Additional Options

The table below lists the additional flags of the command.

Table 47 - Additional ib_read_lat Flags and Options

Flag	Description
--inline_recv=<size>	Max size of message to be sent in inline receive
--output=<units>	Set verbosity output level: bandwidth , message_rate, latency_typical
--pkey_index=<pkey index>	PKey index to use for QP

9.5.3 ib_send_bw

ib_send_bw calculates the BW of SEND between a pair of machines. One acts as a server and the other as a client. The server receive packets from the client and they both calculate the throughput of the operation. The test supports features such as Bidirectional, on which they both send and receive at the same time, change of mtu size, tx size, number of iteration, message size and more. Using the "-a" provides results for all message sizes.

Synopsis

Server

```
ib_send_bw [options]
```

Client

```
ib_send_bw [options] <hostname>
```

Options

The table below lists the various flags of the command.

Table 48 - ib_send_bw Flags and Options

Flag	Description
-a, --all	Run sizes from 2 till 2^23
-b, --bidirectional	Measure bidirectional bandwidth (default unidirectional)
-c, --connection=<RC/XRC/UC/ UD/DC>	Connection type RC/XRC/UC/UD/DC (default RC)
-d, --ib-dev=<dev>	Use IB device <dev> (default first device found)
-D, --duration	Run test for a customized period of seconds.
-e, --events	Sleep on CQ events (default poll)
-f, --margin	Measure results within margins. (default=2sec)
-F, --CPU-freq	Do not fail even if cpufreq_ondemand module is loaded
-g, --mcg=<num_of_qps>	Send messages to multicast group with <num_of_qps> qps attached to it.
-h, --help	Show this help screen.
-i, --ib-port=<port>	Use port <port> of IB device (default 1)
-I, --inline_size=<size>	Max size of message to be sent in inline
-l, --post_list=<list size>	Post list of WQEs of <list size> size (instead of single post)
-m, --mtu=<mtu>	MTU size : 256 - 4096 (default port mtu)
-M, --MGID=<multicast_gid>	In multicast, uses <multicast_gid> as the group MGID.
-n, --iters=<iters>	Number of exchanges (at least 5, default 1000)
-N, --no peak-bw	Cancel peak-bw calculation (default with peak)
-O, --dualport	Run test in dual-port mode.
-p, --port=<port>	Listen on/connect to port <port> (default 18515)
-q, --qp=<num of qp's>	Num of qp's(default 1)

Table 48 - ib_send_bw Flags and Options

Flag	Description
-Q, --cq-mod	Generate Cqe only after <--cq-mod> completion
-r, --rx-depth=<dep>	Rx queue size (default 512). If using srq, rx-depth controls max-wr size of the srq
-R, --rdma_cm	Connect QPs with rdma_cm and run test on those QPs
-s, --size=<size>	Size of message to exchange (default 65536)
-S, --sl=<sl>	SL (default 0)
-t, --tx-depth=<dep>	Size of tx queue (default 128)
-T, --tos=<tos value>	Set <tos_value> to RDMA-CM QPs. available only with -R flag. values 0-256 (default off)
-u, --qp-timeout=<timeout>	QP timeout, timeout value is 4 usec * 2 ^(timeout), default 14
-V, --version	Display version number
-w, --limit_bw	Set verifier limit for bandwidth
-x, --gid-index=<index>	Test uses GID with GID index (Default : IB - no gid . ETH - 0)
-y, --limit_msgrate	Set verifier limit for Msg Rate
-z, --com_rdma_cm	Communicate with rdma_cm module to exchange data - use regular QPs

Additional Options

The table below lists the additional flags of the command.

Table 49 - Additional ib_send_bw Flags and Options

Flag	Description
--inline_recv=<size>	Max size of message to be sent in inline receive
--output=<units>	Set verbosity output level: bandwidth , message_rate, latency_typical
--pkey_index=<pkey index>	PKey index to use for QP
--report-both	Report RX & TX results separately on Bidirectinal BW tests
--report_gbits	Report Max/Average BW of test in Gbit/sec (instead of MB/sec)
--run_infinitely	Run test forever, print results every <duration> seconds

Rate Limiter

The table below lists the Rate Limiter flags of the command.

Table 50 - Additional Rate Limiter Flags and Options

Flag	Description
--burst_size=<size>	Set the amount of messages to send in a burst when using rate limiter
--rate_limit=<rate[pps]>	Set the maximum rate of sent packages
--rate_units=<units> [Mgp]	Set the units for rate limit to MBps (M), Gbps (g) or pps (p)

9.5.4 ib_send_lat

ib_send_lat calculates the latency of sending a packet in message_size between a pair of machines. One acts as a server and the other as a client. They perform a ping pong benchmark on which you send packet only if you receive one. Each of the sides samples the CPU each time they receive a packet in order to calculate the latency. Using the "-a" provides results for all message sizes.

Synopsis

Server

```
ib_send_lat [options]
```

Client

```
ib_send_lat [options] <hostname>
```

Options

The table below lists the various flags of the command.

Table 51 - ib_send_lat Flags and Options

Flag	Description
-a, --all	Run sizes from 2 till 2^23
-c, --connection=<RC/XRC/UC/UD/DC>	Connection type RC/XRC/UC/UD/DC (default RC)
-C, --report-cycles	Report times in cpu cycle units (default microseconds)
-d, --ib-dev=<dev>	Use IB device <dev> (default first device found)
-D, --duration	Run test for a customized period of seconds.
-e, --events	Sleep on CQ events (default poll)
-f, --margin	Measure results within margins. (default=2sec)

Table 51 - ib_send_lat Flags and Options

Flag	Description
-g, --mcg=<num_of_qps>	Send messages to multicast group with <num_of_qps> qps attached to it.
-h, --help	Show this help screen.
-H, --report-histogram	Print out all results (default print summary only)
-i, --ib-port=<port>	Use port <port> of IB device (default 1)
-I, --inline_size=<size>	Max size of message to be sent in inline
-m, --mtu=<mtu>	MTU size : 256 - 4096 (default port mtu)
-M, --MGID=<multicast_gid>	In multicast, uses <multicast_gid> as the group MGID.
-n, --iters=<iters>	Number of exchanges (at least 5, default 1000)
-p, --port=<port>	Listen on/connect to port <port> (default 18515)
-r, --rx-depth=<dep>	Rx queue size (default 512). If using srq, rx-depth controls max-wr size of the srq
-R, --rdma_cm	Connect QPs with rdma_cm and run test on those QPs
-s, --size=<size>	Size of message to exchange (default 2)
-S, --sl=<sl>	SL (default 0)
-T, --tos=<tos value>	Set <tos_value> to RDMA-CM QPs. available only with -R flag. values 0-256 (default off)
-u, --qp-timeout=<timeout>	QP timeout, timeout value is 4 usec * 2 ^(timeout), default 14
-U, --report-unsorted	(implies -H) print out unsorted results (default sorted)
-V, --version	Display version number
-x, --gid-index=<index>	Test uses GID with GID index (Default : IB - no gid . ETH - 0)
-z, --com_rdma_cm	Communicate with rdma_cm module to exchange data - use regular QPs

Additional Options

The table below lists the additional flags of the command.

Table 52 - Additional ib_send_lat Flags and Options

Flag	Description
--inline_recv=<size>	Max size of message to be sent in inline receive

Table 52 - Additional ib_send_lat Flags and Options

Flag	Description
--output=<units>	Set verbosity output level: bandwidth , message_rate, latency_typical
--pkey_index=<pkey index>	PKey index to use for QP

9.5.5 ib_write_bw

ib_write_bw calculates the BW of RDMA write between a pair of machines. One acts as a server and the other as a client. The client RDMA writes to the server memory and calculate the BW by sampling the CPU each time it receive a successful completion. The test supports features such as Bidirectional, in which they both RDMA write to each other at the same time, change of mtu size, tx size, number of iteration, message size and more. Using the "-a" flag provides results for all message sizes.

more. Using the "-a" provides results for all message sizes.

Synopsis

Server

```
ib_write_bw [options]
```

Client

```
ib_write_bw [options] <hostname>
```

Options

The table below lists the various flags of the command.

Table 53 - ib_write_bw Flags and Options

Flag	Description
-a, --all	Run sizes from 2 till 2^23
-b, --bidirectional	Measure bidirectional bandwidth (default unidirectional)
-c, --connection=<RC/XRC/UC/DC>	Connection type RC/XRC/UC/DC (default RC)
-d, --ib-dev=<dev>	Use IB device <dev> (default first device found)
-D, --duration	Run test for a customized period of seconds.
-f, --margin	Measure results within margins. (default=2sec)
-F, --CPU-freq	Do not fail even if cpufreq_ondemand module is loaded
-h, --help	Show this help screen.
-i, --ib-port=<port>	Use port <port> of IB device (default 1)
-I, --inline_size=<size>	Max size of message to be sent in inline

Table 53 - ib_write_bw Flags and Options

Flag	Description
-l, --post_list=<list size>	Post list of WQEs of <list size> size (instead of single post)
-m, --mtu=<mtu>	MTU size : 256 - 4096 (default port mtu)
-n, --iters=<iters>	Number of exchanges (at least 5, default 5000)
-N, --no peak-bw	Cancel peak-bw calculation (default with peak)
-O, --dualport	Run test in dual-port mode.
-p, --port=<port>	Listen on/connect to port <port> (default 18515)
-q, --qp=<num of qp's>	Num of qp's(default 1)
-Q, --cq-mod	Generate Cqe only after <--cq-mod> completion
-R, --rdma_cm	Connect QPs with rdma_cm and run test on those QPs
-s, --size=<size>	Size of message to exchange (default 65536)
-S, --sl=<sl>	SL (default 0)
-t, --tx-depth=<dep>	Size of tx queue (default 128)
-T, --tos=<tos value>	Set <tos_value> to RDMA-CM QPs. available only with -R flag. values 0-256 (default off)
-u, --qp-timeout=<timeout>	QP timeout, timeout value is 4 usec * 2 ^(timeout), default 14
-V, --version	Display version number
-w, --limit_bw	Set verifier limit for bandwidth
-x, --gid-index=<index>	Test uses GID with GID index (Default : IB - no gid . ETH - 0)
-y, --limit_msgrate	Set verifier limit for Msg Rate
-z, --com_rdma_cm	Communicate with rdma_cm module to exchange data - use regular QPs

Additional Options

The table below lists the additional flags of the command.

Table 54 - Additional ib_write_bw Flags and Options

Flag	Description
--output=<units>	Set verbosity output level: bandwidth , message_rate, latency_typical

Table 54 - Additional ib_write_bw Flags and Options

Flag	Description
--pkey_index=<pkey index>	PKey index to use for QP
--report-both	Report RX & TX results separately on Bidirectinal BW tests
--report_gbits	Report Max/Average BW of test in Gbit/sec (instead of MB/sec)
--run_infinitely	Run test forever, print results every <duration> seconds

9.5.6 ib_write_lat

ib_write_lat calculates the latency of RDMA write operation of message_size between a pair of machines. One acts as a server and the other as a client. They perform a ping pong benchmark on which one side RDMA writes to the other side memory only after the other side wrote on his memory. Each of the sides samples the CPU clock each time they write to the other side memory, in order to calculate latency.

Synopsis

Server

```
ib_write_lat [options]
```

Client

```
ib_write_lat [options] <hostname>
```

Options

The table below lists the various flags of the command.

Table 55 - ib_write_lat Flags and Options

Flag	Description
-a, --all	Run sizes from 2 till 2^23
-c, --connection=<RC/XRC/UC/DC>	Connection type RC/XRC/UC/DC (default RC)
-C, --report-cycles	Report times in cpu cycle units (default microseconds)
-d, --ib-dev=<dev>	Use IB device <dev> (default first device found)
-D, --duration	Run test for a customized period of seconds.
-f, --margin	Measure results within margins. (default=2sec)
-h, --help	Show this help screen.
-H, --report-histogram	Print out all results (default print summary only)
-i, --ib-port=<port>	Use port <port> of IB device (default 1)

Table 55 - ib_write_lat Flags and Options

Flag	Description
-I, --inline_size=<size>	Max size of message to be sent in inline
-m, --mtu=<mtu>	MTU size : 256 - 4096 (default port mtu)
-n, --iters=<iters>	Number of exchanges (at least 5, default 1000)
-p, --port=<port>	Listen on/connect to port <port> (default 18515)
-R, --rdma_cm	Connect QPs with rdma_cm and run test on those QPs
-s, --size=<size>	Size of message to exchange (default 2)
-S, --sl=<sl>	SL (default 0)
-T, --tos=<tos value>	Set <tos_value> to RDMA-CM QPs. available only with -R flag. values 0-256 (default off)
-u, --qp-timeout=<timeout>	QP timeout, timeout value is 4 usec * 2 ^(timeout), default 14
-U, --report-unsorted	(implies -H) print out unsorted results (default sorted)
-V, --version	Display version number
-x, --gid-index=<index>	Test uses GID with GID index (Default : IB - no gid . ETH - 0)
-z, --com_rdma_cm	Communicate with rdma_cm module to exchange data - use regular QPs

Additional Options

The table below lists the additional flags of the command.

Table 56 - Additional ib_write_lat Flags and Options

Flag	Description
--output=<units>	Set verbosity output level: bandwidth , message_rate, latency_typical
--pkey_index=<pkey index>	PKey index to use for QP

9.5.7 ib_atomic_bw

Calculates the BW of RDMA Atomic transactions between a pair of machines. One acts as a server and the other as a client. The client RDMA sends atomic operation to the server and calculate the BW by sampling the CPU each time it receive a successful completion. The test supports features such as Bidirectional, in which they both RDMA atomic to each other at the same time, change of MTU size, tx size, number of iteration, message size and more. Using the "-a" flag provides results for all message sizes.

Synopsis

Server

```
ib_atomic_bw [options]
```

Client

```
ib_atomic_bw [options] <hostname>
```

Options

The table below lists the various flags of the command.

Table 57 - ib_atomic_bw Flags and Options

Flag	Description
-A, --atomic_type=<type>	Type of atomic operation from {CMP_AND_SWAP,FETCH_AND_ADD} (default FETCH_AND_ADD)
-b, --bidirectional	Measure bidirectional bandwidth (default unidirectional)
-c, --connection=<RC/XRC/DC>	Connection type RC/XRC/DC (default RC)
-d, --ib-dev=<dev>	Use IB device <dev> (default first device found)
-D, --duration	Run test for a customized period of seconds.
-e, --events	Sleep on CQ events (default poll)
-f, --margin	Measure results within margins. (default=2sec)
-F, --CPU-freq	Do not fail even if cpufreq_ondemand module is loaded
-h, --help	Show this help screen.
-i, --ib-port=<port>	Use port <port> of IB device (default 1)
-l, --post_list=<list size>	Post list of WQEs of <list size> size (instead of single post)
-m, --mtu=<mtu>	MTU size : 256 - 4096 (default port mtu)
-n, --iters=<iters>	Number of exchanges (at least 5, default 1000)
-N, --no peak-bw	Cancel peak-bw calculation (default with peak)
-o, --outs=<num>	Num of outstanding read/atom (default max of device)
-O, --dualport	Run test in dual-port mode.
-p, --port=<port>	Listen on/connect to port <port> (default 18515)
-q, --qp=<num of qp's>	Num of qp's(default 1)
-Q, --cq-mod	Generate Cqe only after <--cq-mod> completion

Table 57 - ib_atomic_bw Flags and Options

Flag	Description
-R, --rdma_cm	Connect QPs with rdma_cm and run test on those QPs
-S, --sl=<sl>	SL (default 0)
-t, --tx-depth=<dep>	Size of tx queue (default 128)
-T, --tos=<tos value>	Set <tos_value> to RDMA-CM QPs. available only with -R flag. values 0-256 (default off)
-u, --qp-timeout=<timeout>	QP timeout, timeout value is 4 usec * 2 ^(timeout), default 14
-V, --version	Display version number
-w, --limit_bw	Set verifier limit for bandwidth
-x, --gid-index=<index>	Test uses GID with GID index (Default : IB - no gid . ETH - 0)
-y, --limit_msgrate	Set verifier limit for Msg Rate
-z, --com_rdma_cm	Communicate with rdma_cm module to exchange data - use regular QPs

Additional Options

The table below lists the additional flags of the command.

Table 58 - Additional ib_atomic_bw Flags and Options

Flag	Description
--inline_recv=<size>	Max size of message to be sent in inline receive
--output=<units>	Set verbosity output level: bandwidth , message_rate, latency_typical
--pkey_index=<pkey index>	PKey index to use for QP
--report-both	Report RX & TX results separately on Bidirectinal BW tests
--report_gbits	Report Max/Average BW of test in Gbit/sec (instead of MB/sec)
--run_infinitely	Run test forever, print results every <duration> seconds

9.5.8 ib_atomic_lat

Calculates the latency of RDMA Atomic transaction of message_size between a pair of machines. One acts as a server and the other as a client. The client sends RDMA atomic operation and sample the CPU clock when it receives a successful completion, in order to calculate latency.

Synopsis

Server

```
ib_atomic_lat [options]
```

Client

```
ib_atomic_lat [options] <hostname>
```

Options

The table below lists the various flags of the command.

Table 59 - ib_atomic_lat Flags and Options

Flag	Description
-A, --atomic_type=<type>	Type of atomic operation from {CMP_AND_SWAP,FETCH_AND_ADD} (default FETCH_AND_ADD)
-c, --connection=<RC/XRC/DC>	Connection type RC/XRC/DC (default RC)
-C, --report-cycles	Report times in cpu cycle units (default microseconds)
-d, --ib-dev=<dev>	Use IB device <dev> (default first device found)
-D, --duration	Run test for a customized period of seconds.
-e, --events	Sleep on CQ events (default poll)
-f, --margin	Measure results within margins. (default=2sec)
-h, --help	Show this help screen.
-H, --report-histogram	Print out all results (default print summary only)
-i, --ib-port=<port>	Use port <port> of IB device (default 1)
-m, --mtu=<mtu>	MTU size : 256 - 4096 (default port mtu)
-n, --iters=<iters>	Number of exchanges (at least 5, default 1000)
-o, --outs=<num>	Num of outstanding read/atom(default max of device)
-p, --port=<port>	Listen on/connect to port <port> (default 18515)
-R, --rdma_cm	Connect QPs with rdma_cm and run test on those QPs
-S, --sl=<sl>	SL (default 0)
-T, --tos=<tos value>	Set <tos_value> to RDMA-CM QPs. availible only with -R flag. values 0-256 (default off)
-u, --qp-timeout=<timeout>	QP timeout, timeout value is 4 usec * 2 ^(timeout), default 14
-U, --report-unsorted	(implies -H) print out unsorted results (default sorted)

Table 59 - ib_atomic_lat Flags and Options

Flag	Description
-V, --version	Display version number
-x, --gid-index=<index>	Test uses GID with GID index (Default : IB - no gid . ETH - 0)
-z, --com_rdma_cm	Communicate with rdma_cm module to exchange data - use regular QPs

Additional Options

The table below lists the additional flags of the command.

Table 60 - Additional ib_atomic_lat Flags and Options

Flag	Description
--inline_recv=<size>	Max size of message to be sent in inline receive
--output=<units>	Set verbosity output level: bandwidth, message_rate, latency_typical
--pkey_index=<pkey index>	PKey index to use for QP

9.5.9 raw_ethernet_bw

raw_ethernet_bw calculates the BW of SEND between a pair of machines. One acts as a server and the other as a client. The server receive packets from the client and they both calculate the throughput of the operation. The test supports features such as Bidirectional, on which they both send and receive at the same time, change of mtu size, tx size, number of iteration, message size and more. Using the "-a" provides results for all message sizes.

Server

```
raw_ethernet_bw [options]
```

Client

```
raw_ethernet_bw [options] --dest_mac=<MAC address of interface on server>
```

Options

The table below lists the various flags of the command.

Table 61 - raw_ethernet_bw Flags and Options

Flag	Description
-a, --all	Run sizes from 2 till 2^23
-b, --bidirectional	Measure bidirectional bandwidth (default unidirectional)
-c, --connection=<RC/XRC/UC/UD/DC>	Connection type RC/XRC/UC/UD/DC (default RC)

Table 61 - raw_ethernet_bw Flags and Options

Flag	Description
-d, --ib-dev=<dev>	Use IB device <dev> (default first device found)
-D, --duration	Run test for a customized period of seconds.
-e, --events	Sleep on CQ events (default poll)
-f, --margin	Measure results within margins. (default=2sec)
-F, --CPU-freq	Do not fail even if cpufreq_ondemand module is loaded
-g, --mcg=<num_of_qps>	Send messages to multicast group with <num_of_qps> qps attached to it.
-h, --help	Show this help screen.
-i, --ib-port=<port>	Use port <port> of IB device (default 1)
-I, --inline_size=<size>	Max size of message to be sent in inline
-l, --post_list=<list size>	Post list of WQEs of <list size> size (instead of single post)
-m, --mtu=<mtu>	MTU size : 256 - 4096 (default port mtu)
-M, --MGID=<multicast_gid>	In multicast, uses <multicast_gid> as the group MGID.
-n, --iters=<iters>	Number of exchanges (at least 5, default 1000)
-N, --no peak-bw	Cancel peak-bw calculation (default with peak)
-O, --dualport	Run test in dual-port mode.
-p, --port=<port>	Listen on/connect to port <port> (default 18515)
-q, --qp=<num of qp's>	Num of qp's(default 1)
-Q, --cq-mod	Generate Cqe only after <--cq-mod> completion
-r, --rx-depth=<dep>	Rx queue size (default 512). If using srq, rx-depth controls max-wr size of the srq
-R, --rdma_cm	Connect QPs with rdma_cm and run test on those QPs
-s, --size=<size>	Size of message to exchange (default 65536)
-S, --sl=<sl>	SL (default 0)
-t, --tx-depth=<dep>	Size of tx queue (default 128)
-T, --tos=<tos value>	Set <tos_value> to RDMA-CM QPs. available only with -R flag. values 0-256 (default off)
-u, --qp-timeout=<timeout>	QP timeout, timeout value is 4 usec * 2 ^(timeout), default 14

Table 61 - raw_ethernet_bw Flags and Options

Flag	Description
-V, --version	Display version number
-w, --limit_bw	Set verifier limit for bandwidth
-x, --gid-index=<index>	Test uses GID with GID index (Default : IB - no gid . ETH - 0)
-y, --limit_msgrate	Set verifier limit for Msg Rate
-z, --com_rdma_cm	Communicate with rdma_cm module to exchange data - use regular QPs

Additional Options

The table below lists the additional flags of the command.

Table 62 - Additional raw_ethernet_bw Flags and Options

Flag	Description
--inline_recv=<size>	Max size of message to be sent in inline receive
--output=<units>	Set verbosity output level: bandwidth , message_rate, latency_typical
--pkey_index=<pkey index>	PKey index to use for QP
--report-both	Report RX & TX results separately on Bidirectinal BW tests
--report_gbits	Report Max/Average BW of test in Gbit/sec (instead of MB/sec)
--run_infinitely	Run test forever, print results every <duration> seconds

Rate Limiter

The table below lists the Rate Limiter flags of the command.

Table 63 - raw_ethernet_bw Rate Limiter Flags and Options

Flag	Description
--burst_size=<size>	Set the amount of messages to send in a burst when using rate limiter
--rate_limit=<rate[pps]>	Set the maximum rate of sent packages
--rate_units=<units> [Mgp]	Set the units for rate limit to MBps (M), Gbps (g) or pps (p)

Raw Ethernet Options

The table below lists the Raw Ethernet flags of the command.

Table 64 - raw_ethernet_bw Raw Ethernet Flags and Options

Flag	Description
-B, --source_mac	Source MAC address by this format XX:XX:XX:XX:XX:XX (default take the MAC address form GID)
-E, --dest_mac	Destination MAC address by this format XX:XX:XX:XX:XX:XX **MUST** be entered
-J, --dest_ip	Destination ip address by this format X.X.X.X (using to send packets with IP header)
-j, --source_ip	Source ip address by this format X.X.X.X (using to send packets with IP header)
-K, --dest_port	Destination port number (using to send packets with UDP header as default, or you can use --tcp flag to send TCP Header)
-k, --source_port	Source port number (using to send packets with UDP header as default, or you can use --tcp flag to send TCP Header)
-Z, --server	Choose server side for the current machine (--server/--client must be selected)
-P, --client	Choose client side for the current machine (--server/--client must be selected)
-v, --mac_fwd	Run mac forwarding test
--tcp	Send TCP Packets. must include IP and Ports information.

9.5.10 raw_ethernet_lat

raw_ethernet_lat calculates the latency of sending a packet in message_size between a pair of machines. One acts as a server and the other as a client. They perform a ping pong benchmark on which you send packet only if you receive one. Each of the sides samples the CPU each time they receive a packet in order to calculate the latency. Using the "-a" provides results for all message sizes.

Synopsis

Server

```
raw_ethernet_lat [options] --dest_mac=<MAC address of interface on client>
```

Client

```
raw_ethernet_lat [options] --dest_mac=<MAC address of interface on server>
```

Options

The table below lists the various flags of the command.

Table 65 - raw_ethernet_lat Flags and Options

Flag	Description
-a, --all	Run sizes from 2 till 2^23
-c, --connection=<RC/XRC/UC/ UD/DC>	Connection type RC/XRC/UC/UD/DC (default RC)
-C, --report-cycles	Report times in cpu cycle units (default microseconds)
-d, --ib-dev=<dev>	Use IB device <dev> (default first device found)
-D, --duration	Run test for a customized period of seconds.
-e, --events	Sleep on CQ events (default poll)
-f, --margin	Measure results within margins. (default=2sec)
-g, --mcg=<num_of_qps>	Send messages to multicast group with <num_of_qps> qps attached to it.
-h, --help	Show this help screen.
-H, --report-histogram	Print out all results (default print summary only)
-i, --ib-port=<port>	Use port <port> of IB device (default 1)
-I, --inline_size=<size>	Max size of message to be sent in inline
-m, --mtu=<mtu>	MTU size : 256 - 4096 (default port mtu)
-M, --MGID=<multicast_gid>	In multicast, uses <multicast_gid> as the group MGID.
-n, --iters=<iters>	Number of exchanges (at least 5, default 1000)
-p, --port=<port>	Listen on/connect to port <port> (default 18515)

Table 65 - raw_ethernet_lat Flags and Options

Flag	Description
-r, --rx-depth=<dep>	Rx queue size (default 512). If using srq, rx-depth controls max-wr size of the srq
-R, --rdma_cm	Connect QPs with rdma_cm and run test on those QPs
-s, --size=<size>	Size of message to exchange (default 2)
-S, --sl=<sl>	SL (default 0)
-T, --tos=<tos value>	Set <tos_value> to RDMA-CM QPs. availible only with -R flag. values 0-256 (default off)
-u, --qp-timeout=<timeout>	QP timeout, timeout value is 4 usec * 2 ^(timeout), default 14
-U, --report-unsorted	(implies -H) print out unsorted results (default sorted)
-V, --version	Display version number
-x, --gid-index=<index>	Test uses GID with GID index (Default : IB - no gid . ETH - 0)
-z, --com_rdma_cm	Communicate with rdma_cm module to exchange data - use regular QPs

Additional Options

The table below lists the additional flags of the command.

Table 66 - Additional raw_ethernet_lat Flags and Options

Flag	Description
--inline_recv=<size>	Max size of message to be sent in inline receive
--output=<units>	Set verbosity output level: bandwidth , message_rate, latency_typical
--pkey_index=<pkey index>	PKey index to use for QP

Raw Ethernet Options

The table below lists the Raw Ethernet flags of the command.

Table 67 - raw_ethernet_lat Raw Ethernet Flags and Options

Flag	Description
-B, --source_mac	Source MAC address by this format XX:XX:XX:XX:XX:XX (default take the MAC address form GID)

Table 67 - raw_ethernet_lat Raw Ethernet Flags and Options

Flag	Description
-E, --dest_mac	Destination MAC address by this format XX:XX:XX:XX:XX:XX **MUST** be entered
-J, --dest_ip	Destination ip address by this format X.X.X.X (using to send packets with IP header)
-j, --source_ip	Source ip address by this format X.X.X.X (using to send packets with IP header)
-K, --dest_port	Destination port number (using to send packets with UDP header as default, or you can use --tcp flag to send TCP Header)
-k, --source_port	Source port number (using to send packets with UDP header as default, or you can use --tcp flag to send TCP Header)
-Z, --server	Choose server side for the current machine (--server/--client must be selected)
-P, --client	Choose client side for the current machine (--server/--client must be selected)
-v, --mac_fwd	Run mac forwarding test
--tcp	Send TCP Packets. must include IP and Ports information.

Appendix A: SRP Target Driver

The SRP Target driver is designed to work directly on top of OpenFabrics OFED software stacks (http://www.openfabrics.org) or InfiniBand drivers in Linux kernel tree (kernel.org). It also interfaces with Generic SCSI target mid-level driver - SCST (http://scst.sourceforge.net).

By interfacing with an SCST driver, it is possible to work with and support a lot of IO modes on real or virtual devices in the back end.

1. scst_vdisk – fileio and blockio modes. This allows turning software raid volumes, LVM volumes, IDE disks, block devices and normal files into SRP luns
2. NULLIO mode allows measuring the performance without sending IOs to *real* devices

A.1 Prerequisites and Installation

1. SRP target is part of the OpenFabrics OFED software stacks. Use the latest OFED distribution package to install SRP target.

On distribution default kernels you can run scst_vdisk blockio mode to obtain good performance.

2. Download and install the SCST driver. The supported version is 1.0.1.1.
   a. Download `scst-1.0.1.1.tar.gz` from http://scst.sourceforge.net/downloads.html
   b. Untar `scst-1.0.1.1`

```
$ tar zxvf scst-1.0.1.1.tar.gz
$ cd scst-1.0.1.1
```

   c. Install `scst-1.0.1.1 as follows:`

```
$ make && make install
```

A.2 How-to Run

A. On an SRP Target machine:

1. Please refer to SCST's README for loading scst driver and its dev_handlers drivers (scst_vdisk block or file IO mode, nullio, ...)

Regardless of the mode, you always need to have lun 0 in any group's device list. Then you can have any lun number following lun 0 (it is not required to have the lun numbers in ascending order except that the first lun must always be 0).

Setting SRPT_LOAD=yes in /etc/infiniband/openib.conf is not enough as it only loads the `ib_srpt` module but does *not* load `scst` not its dev_handlers.

The scst_disk module (pass-thru mode) of SCST is not supported by Mellanox OFED.

Example 1: Working with VDISK BLOCKIO mode

(Using the md0 device, sda, and cciss/c1d0)

a. modprobe scst
b. modprobe scst_vdisk
c. echo "open vdisk0 /dev/md0 BLOCKIO" > /proc/scsi_tgt/vdisk/vdisk
d. echo "open vdisk1 /dev/sda BLOCKIO" > /proc/scsi_tgt/vdisk/vdisk
e. echo "open vdisk2 /dev/cciss/c1d0 BLOCKIO" > /proc/scsi_tgt/vdisk/vdisk
f. echo "add vdisk0 0" >/proc/scsi_tgt/groups/Default/devices
g. echo "add vdisk1 1" >/proc/scsi_tgt/groups/Default/devices
h. echo "add vdisk2 2" >/proc/scsi_tgt/groups/Default/devices

Example 2: working with scst_vdisk FILEIO mode

(Using md0 device and file 10G-file)

a. modprobe scst
b. modprobe scst_vdisk
c. echo "open vdisk0 /dev/md0" > /proc/scsi_tgt/vdisk/vdisk
d. echo "open vdisk1 /10G-file" > /proc/scsi_tgt/vdisk/vdisk
e. echo "add vdisk0 0" >/proc/scsi_tgt/groups/Default/devices
f. echo "add vdisk1 1" >/proc/scsi_tgt/groups/Default/devices

2. Run:

```
For all distributions except SLES 11:  > modprobe ib_srpt
For SLES 11:   > modprobe -f ib_srpt
```

For SLES 11, please ignore the following error messages in /var/log/messages when loading ib_srpt to SLES 11 distribution's kernel:

```
...
ib_srpt: no symbol version for scst_unregister
ib_srpt: Unknown symbol scst_unregister
ib_srpt: no symbol version for scst_register
ib_srpt: Unknown symbol scst_register
ib_srpt: no symbol version for scst_unregister_target_template
ib_srpt: Unknown symbol scst_unregister_target_template
...
```

B. On Initiator Machines

On Initiator machines, manually perform the following steps:

1. Run:

```
modprobe ib_srp
```

2. Run:

```
ibsrpdm -c -d /dev/infiniband/umadX
```

(to discover a new SRP target)

```
umad0: port 1 of the first HCA
```

```
umad1: port 2 of the first HCA
umad2: port 1 of the second HCA
```

3. echo {new target info} > /sys/class/infiniband_srp/srp-mthca0-1/add_target
4. fdisk -l (will show the newly discovered scsi disks)

Example:

Assume that you use port 1 of first HCA in the system, i.e.: mthca0

```
[root@lab104 ~]# ibsrpdm -c -d /dev/infiniband/umad0
id_ext=0002c90200226cf4,ioc_guid=0002c90200226cf4,
dgid=fe8000000000000002c90200226cf5,pkey=ffff,service_id=0002c90200226cf4
[root@lab104 ~]# echo id_ext=0002c90200226cf4,ioc_guid=0002c90200226cf4,
dgid=fe8000000000000002c90200226cf5,pkey=ffff,service_id=0002c90200226cf4 > /sys/
class/infiniband_srp/srp-mthca0-1/add_target
```

OR

- You can edit /etc/infiniband/openib.conf to load the SRP driver and SRP High Availability (HA) daemon automatically, that is: set "SRP_LOAD=yes" and "SRPHA_ENABLE=yes"
- To set up and use the HA feature, you need the dm-multipath driver and multipath tool
- Please refer to OFED-1.x SRP's user manual for more detailed instructions on how-to enable/use the HA feature

The following is an example of an SRP Target setup file:

```
********************** srpt.sh *******************************
#!/bin/sh
modprobe scst scst_threads=1
modprobe scst_vdisk scst_vdisk_ID=100

echo "open vdisk0 /dev/cciss/c1d0 BLOCKIO" > /proc/scsi_tgt/vdisk/vdisk
echo "open vdisk1 /dev/sdb BLOCKIO" > /proc/scsi_tgt/vdisk/vdisk
echo "open vdisk2 /dev/sdc BLOCKIO" > /proc/scsi_tgt/vdisk/vdisk
echo "open vdisk3 /dev/sdd BLOCKIO" > /proc/scsi_tgt/vdisk/vdisk
echo "add vdisk0 0" > /proc/scsi_tgt/groups/Default/devices
echo "add vdisk1 1" > /proc/scsi_tgt/groups/Default/devices
echo "add vdisk2 2" > /proc/scsi_tgt/groups/Default/devices
echo "add vdisk3 3" > /proc/scsi_tgt/groups/Default/devices

modprobe ib_srpt

echo "add "mgmt"" > /proc/scsi_tgt/trace_level
echo "add "mgmt_dbg"" > /proc/scsi_tgt/trace_level
echo "add "out_of_mem"" > /proc/scsi_tgt/trace_level
********************** End srpt.sh ****************************
```

A.3 How-to Unload/Shutdown

1. Unload `ib_srpt`.

```
$ modprobe -r ib_srpt
```

2. Unload `scst` and its dev_handlers first.

```
$ modprobe -r scst_vdisk scst
```

3. Unload MLNX_OFED kernel modules.

```
$ /etc/rc.d/openibd stop
```

Appendix B: mlx4 Module Parameters

In order to set **mlx4** parameters, add the following line(s) to **/etc/modprobe.conf**:

```
options mlx4_core parameter=<value>
```

and/or

```
options mlx4_ib   parameter=<value>
```

and/or

```
options mlx4_en   parameter=<value>
```

The following sections list the available mlx4 parameters.

B.1 mlx4_ib Parameters

sm_guid_assign:	Enable SM alias_GUID assignment if sm_guid_assign > 0 (Default: 1) (int)
dev_assign_str[1]:	Map device function numbers to IB device numbers (e.g.'0000:04:00.0-0,002b:1c:0b.a-1,...'). Hexadecimal digits for the device function (e.g. 002b:1c:0b.a) and decimal for IB device numbers (e.g. 1). Max supported devices - 32 (string)

1. In the current version, this parameter is using decimal number to describe the InfiniBand device and not hexadecimal number as it was in previous versions in order to uniform the mapping of device function numbers to InfiniBand device numbers as defined for other module parameters (e.g. num_vfs and probe_vf).
   For example to map mlx4_15 to device function number 04:00.0 in the current version we use "options mlx4_ib dev_assign_str=04:00.0-15" as opposed to the previous version in which we used "options mlx4_ib dev_assign_str=04:00.0-f"

B.2 mlx4_core Parameters

set_4k_mtu:	(Obsolete) attempt to set 4K MTU to all ConnectX ports (int)
debug_level:	Enable debug tracing if > 0 (int)
msi_x:	0 - don't use MSI-X, 1 - use MSI-X, >1 - limit number of MSI-X irqs to msi_x (non-SRIOV only) (int)
enable_sys_tune:	Tune the cpu's for better performance (default 0) (int)
block_loopback:	Block multicast loopback packets if > 0 (default: 1) (int)
num_vfs:	Either a single value (e.g. '5') to define uniform num_vfs value for all devices functions or a string to map device function numbers to their num_vfs values (e.g. '0000:04:00.0-5,002b:1c:0b.a-15'). Hexadecimal digits for the device function (e.g. 002b:1c:0b.a) and decimal for num_vfs value (e.g. 15). (string)
probe_vf:	Either a single value (e.g. '3') to indicate that the Hypervisor driver itself should activate this number of VFs for each HCA on the host, or a string to map device function numbers to their probe_vf values (e.g. '0000:04:00.0-3,002b:1c:0b.a-13'). Hexadecimal digits for the device function (e.g. 002b:1c:0b.a) and decimal for probe_vf value (e.g. 13). (string)

log_num_mgm_entry_size:	log mgm size, that defines the num of qp per mcg, for example: 10 gives 248.range: 7 <= log_num_mgm_entry_size <= 12. To activate device managed flow steering when available, set to -1 (int)
high_rate_steer:	Enable steering mode for higher packet rate (default off) (int)
fast_drop:	Enable fast packet drop when no recieve WQEs are posted (int)
enable_64b_cqe_eqe:	Enable 64 byte CQEs/EQEs when the FW supports this if non-zero (default: 1) (int)
log_num_mac:	Log2 max number of MACs per ETH port (1-7) (int)
log_num_vlan:	(Obsolete) Log2 max number of VLANs per ETH port (0-7) (int)
log_mtts_per_seg:	Log2 number of MTT entries per segment (0-7) (default: 0) (int)
port_type_array:	Either pair of values (e.g. '1,2') to define uniform port1/ port2 types configuration for all devices functions or a string to map device function numbers to their pair of port types values (e.g. '0000:04:00.0-1;2,002b:1c:0b.a-1;1'). Valid port types: 1-ib, 2-eth, 3-auto, 4-N/A If only a single port is available, use the N/A port type for port2 (e.g '1,4').
log_num_qp:	log maximum number of QPs per HCA (default: 19) (int)
log_num_srq:	log maximum number of SRQs per HCA (default: 16) (int)
log_rdmarc_per_qp:	log number of RDMARC buffers per QP (default: 4) (int)
log_num_cq:	log maximum number of CQs per HCA (default: 16) (int)
log_num_mcg:	log maximum number of multicast groups per HCA (default: 13) (int)
log_num_mpt:	log maximum number of memory protection table entries per HCA (default: 19) (int)
log_num_mtt:	log maximum number of memory translation table segments per HCA (default: max(20, 2*MTTs for register all of the host memory limited to 30)) (int)
enable_qos:	Enable Quality of Service support in the HCA (default: off) (bool)
internal_err_reset:	Reset device on internal errors if non-zero (default is 0) (int)

B.3 mlx4_en Parameters

inline_thold:	Threshold for using inline data (int) Default and max value is 104 bytes. Saves PCI read operation transaction, packet less then threshold size will be copied to hw buffer directly.
udp_rss:	Enable RSS for incoming UDP traffic (uint) On by default. Once disabled no RSS for incoming UDP traffic will be done.
pfctx:	Priority based Flow Control policy on TX[7:0]. Per priority bit mask (uint)
pfcrx:	Priority based Flow Control policy on RX[7:0]. Per priority bit mask (uint)

Appendix C: mlx5 Module Parameters

The mlx5_ib module supports a single parameter used to select the profile which defines the number of resources supported. The parameter name for selecting the profile is `prof_sel`.

The supported values for profiles are:

- 0 - for medium resources, medium performance
- 1 - for low resources
- 2 - for high performance (int) (default)

Appendix D: Lustre Compilation over MLNX_OFED

This procedure applies to RHEL/SLES OSs supported by Lustre. For further information, please refer to Lustre Release Notes.

- ***To compile Lustre version 2.4.0 and higher:***

```
$ ./configure --with-o2ib=/usr/src/ofa_kernel/default/
$ make rpms
```

- ***To compile older Lustre versions:***

```
$ EXTRA_LNET_INCLUDE="-I/usr/src/ofa_kernel/default/include/ -include /usr/src/
ofa_kernel/default/include/linux/compat-2.6.h" ./configure --with-o2ib=/usr/src/
ofa_kernel/default/
$ EXTRA_LNET_INCLUDE="-I/usr/src/ofa_kernel/default/include/ -include /usr/src/
ofa_kernel/default/include/linux/compat-2.6.h" make rpms
```

Appendix E: Using FlexBoot for Booting Various OSs from an iSCSI Target

Below are instructions on how to provision a diskless system (the client) with a fresh SLES11 SP3 and RHEL6.4 installation to a remote storage (IE: a LUN partition on an iSCSI target) and then SAN-Booting (iSCSI boot) the client using Mellanox PXE boot agent (FlexBoot). The iSCSI configuration in this document is very basic (no CHAP authentication, no multipath I/O) and demonstrates basic PXE SAN Boot capability.

In this chapter, IET is used as the iSCSI Target.

In the example below:

- sqa030 (a Linux host) - performs the role of the PXE server and an iSCSI target and
- sqa070 (a host) - performs the role of the PXE client and a sanity check environment
- sqa030 – The OS is configured with IP 12.7.6.30, and the DHCP IP configuration for the diskless client is 12.7.6.70
- sqa030 - has a 21.5GB disk partition on its local HDD, called /dev/cciss/c0d0p9, which will be used as the storage volume to host the client's OS. It also serves as DHCP, TFTP, and NFS server for PXE clients
- Both servers have a Mellanox ConnectX®-3 based 10GE NIC equipped with PXE boot capabilities via Expansion ROM called FlexBoot (http://www.mellanox.com/page/products_dyn?&product_family=34&mtag=flexboot).

E.1 Configuring the iSCSI Target Machine

➢ ***To configure the iSCSI target:***

Step 1. Download the IET target software from.

http://sourceforge.net/projects/iscsitarget/files/iscsitarget/1.4.20.2/

Step 2. Install iSCSI target and additional required software on target server.

```
[root@sqa030 ~]# yum install kernel-devel openssl-devel gcc rpm-build
[root@sqa030 tmp]# tar xzvf iscsitarget-1.4.20.2.tar.gz
[root@sqa030 tmp]# cd iscsitarget-1.4.20.2/
[root@sqa030 iscsitarget-1.4.20.2]# make && make install
```

Step 3. Create the IQN in the ietd configuration file.

```
Target iqn.2013-10.qalab.com:sqa030.prt9
Lun 0 Path=/dev/cciss/c0d0p9,Type=fileio,IOMode=wb
MaxConnections           1       # Number of connections/session. We only support 1
InitialR2T               Yes     # Wait first for R2T
ImmediateData            Yes     # Data can accompany command
MaxRecvDataSegmentLength  8192   # Max data per PDU to receive
MaxXmitDataSegmentLength  8192   # Max data per PDU to transmit
MaxBurstLength           262144  # Max data per sequence (R2T)
FirstBurstLength         65536   # Max unsolicited data sequence
DefaultTime2Wait         2       # Secs wait for ini to log out   Not used
DefaultTime2Retain       20      # Secs keep cmnds after log out Not used
MaxOutstandingR2T        1       # Max outstanding R2Ts per cmnd
DataPDUInOrder           Yes     # Data in PDUs is ordered. We only support ordered
DataSequenceInOrder      Yes     # PDUs in sequence are ordered. We only support
                                    ordered
ErrorRecoveryLevel       0       # We only support level 0
HeaderDigest             NONE    # PDU header checksum algo list.  None or CRC32C
                                 # If only one is set then the initiator must agree
                                 # to it or the connection will fail
DataDigest               NONE    # PDU data checksum algo list     Same as above
MaxSessions              0       # Maximum number of sessions to this target 0 =
                                    unlimited
NOPInterval              0       # Send a NOP-In ping each after that many seconds
                                    if the
                                 # conn is otherwise idle  0 = off
NOPTimeout               0       # Wait that many seconds for a response on a
                                    NOP-In ping
                                 # If 0 or > NOPInterval, NOPInterval is used!
                                 # Various target parameters
Wthreads                 8       # Number of IO threads
QueuedCommands           32      # Number of queued commands
```

The local Hard Disk partition assigned to the LUN (/dev/cciss/c0d0p9 in the example above) must not contain any valuable data, as this data will be destroyed by the installation process taking place later in this procedure

Step 4. Edit the /etc/sysconfig/iscsi-target file as follow.

```
OPTIONS="-c /etc/iet/ietd.conf --address=12.7.6.30"
```

Step 5. Start the iSCSI target service.

```
[root@sqa030 ~]# /etc/init.d/iscsi-target start
```

Step 6. Perform a sanity check by connecting to the iSCSI target from a remote PC on the 10GE network link.

E.2 Configuring the DHCP Server

Edit a host-declaration for your PXE client in the DHCP configuration file, serving it with pxelinux.0, and restart your DHCP.

"pxelinux.0" is a part of the "syslinux" RPM (/usr/share/syslinux/pxelinux.0).
To install the syslinux RPM on:

- RedHat, run "yum install syslinux"
- SUSE - "zypper install syslinux"

To get the latest version of syslinux please refer to http://www.syslinux.org

Here is an example of such host declaration inside DHCP config file:

```
host qadell011 {
       filename "pxelinux.0" ;
       next-server 12.7.6.30;
       option host-name "qadell011";
       fixed-address 12.7.6.11  ;
       hardware ethernet 00:02:C9:E5:D8:E0 ;
}
```

E.3 Configuring the PXE Server

Step 1. Download SLES11SP3-kISO-VPI.tgz from http://www.mellanox.com/page/products_dyn?&product_family=34&mtag=flexboot

Step 2. Extract the .tgz file on the PXE server, under the TFTP root directory.

For example:

```
[root@sqa030 ~]# cd /var/lib/tftpboot;  tar xzvf SLES11SP3-kISO-VPI.tgz
```

The following are examples of PXE server configuration:

- PXE server configuration:

```
SLES11SP3-kISO-VPI/pxelinux.cfg/default
```

- Kernel and initrd for the installation:

```
SLES11SP3-kISO-VPI/pxeboot-install/initrd
SLES11SP3-kISO-VPI/pxeboot-install/linux
```

- Kernel and initrd for the boot after the installation:

```
SLES11SP3-kISO-VPI/pxeboot/initrd
SLES11SP3-kISO-VPI/pxeboot/linux
```

- kISO installation medium that can be used to boot from instead of booting the installer program over the network.

  If choosing this method:

  - Boot the client into the below SLES11 SP3 iso and proceed with the installation until the client fully boots up the installer program.
  - Discover and connect to a remote iSCSI storage.
    During the installation process you will be asked to insert the original installation medium to continue with the installation.

```
SLES11SP3-kISO-VPI/sles11-sp3-x86_64-mlnx_ofed-2.1-1.0.6.iso
```

If the iso method above is not used, two different PXE server configurations are required (PXELINUX booting labels) for each phase discussed herein (booting the installer and post-installation boot)

- For booting the installer program off the TFTP server, please provide the client a path to the initrd and linux kernel as provided inside SLES11SP3-kISO-VPI/pxeboot-install/ in the tgz above.

  The below is an example of such label.

```
LABEL SLES11.3x64_manual_installl
MENU LABEL ^1) Install SLES11.3
kernel SLES11SP3-kISO-VPI/pxeboot-install/linux
append initrd=SLES11SP3-kISO-VPI/pxeboot-install/initrd install=nfs://12.7.6.30/pxerepo/
SLES/11.3/x86_64/DVD1/?device=p4p2
IPAPPEND 2
```

To install SLES11 SP3 over iSCSI over IPoIB, the following parameters need to be added to the “append” statement above:
insmod=ib_ipoib insmod=libiscsi insmod=rdma_cm insmod=ib_iser

- For post-installation boot (booting the SLES 11 SP3 off the iSCSI storage using PXE services) please provide the booting client a path to the initrd and linux kernel as provided inside SLES11SP3-kISO-VPI/pxeboot/ in the tgz above.

  The below is an example of such label.

```
LABEL SLES11.3x64_iscsi_boot
MENU LABEL ^2) SLES11.3 iSCSI boot
kernel SLES11SP3-kISO-VPI/pxeboot/linux
append initrd= SLES11SP3-kISO-VPI/pxeboot/initrd net-root=iscsi:12.7.6.30::::iqn.2013-
10.qalab.com:sqa030.prt9 TargetAd-dress=12.7.6.30 TargetName=iqn.2013-
10.qalab.com:sqa030.prt9 TargetPort=3260 net_delay=10 rootfstype=ext3 rootdev=/dev/sda2
```

The steps described in this document do not refer to an unattended installation with autoyast. For official information on SLES unattented installation with autoyast, please refer to:

https://www.suse.com/documentation/sles11/book_autoyast/?page=/documentation/sles11/book_autoyast/data/book_autoyast.html

The following is known to work with Mellanox NIC:

```
append initrd=SLES-11-SP3-DVD-x86_64-GM-DVD1/boot/x86_64/loader/initrd install=nfs://
<NFS IP Address>/<path the the repository directory>/ autoyast=nfs://<NFS IP Address>/
<path to autoyast xml directory>/autoyast-unattended.xml biosdevname=0
IPAPPEND 2
```

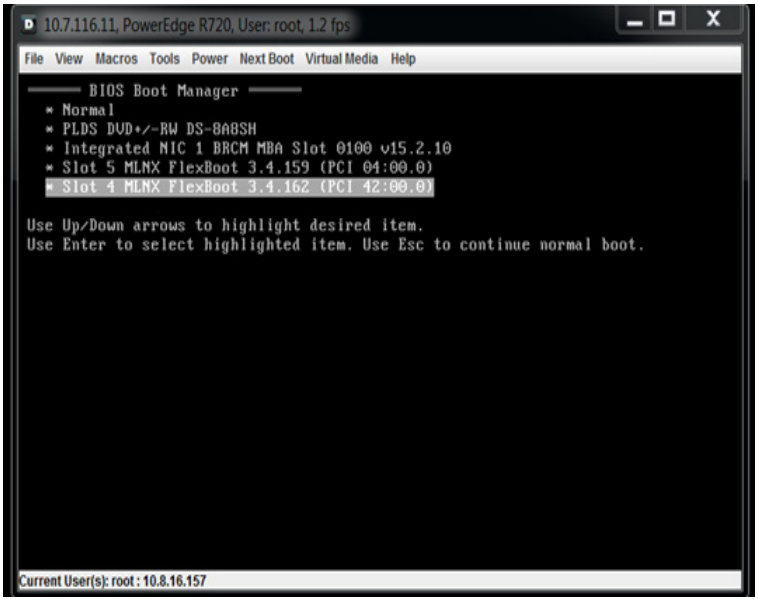
E.4 Installing SLES11 SP3 on a Remote Storage over iSCSI

Step 1. Reboot the diskless client and perform a PXE boot with FlexBoot.

This is not an iSCSI boot, rather a regular PXE initiated network deployment of SLES11 SP3. In the DHCP server configuration, the PXELINUX (pxelinux.0) and a SLES11 SP3 distribution media will be provided for network installation.

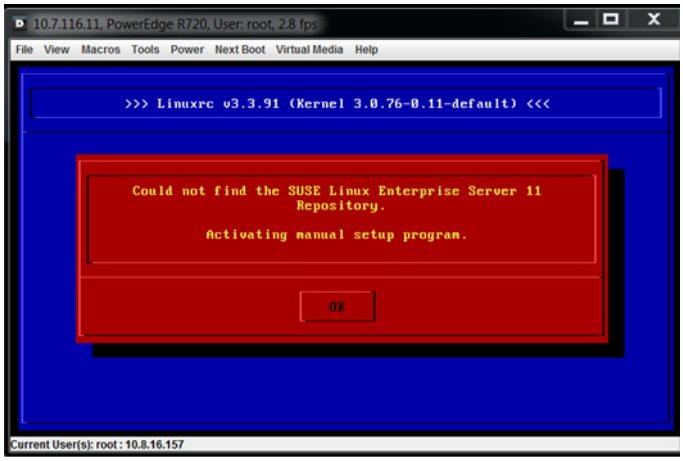
The clients’ HDD was removed beforehand; therefore the installer will ask to locate a HDD. The built-in iSCSI discovery will be used to connect to the iSCSI target LUN partition.

Step 2. Reboot the client and invoke PXE boot with the Mellanox boot agent.


10.7.116.11, PowerEdge R720, User: root, 1.2 fps
File View Macros Tools Power Next Boot Virtual Media Help
BIOS Boot Manager
* Normal
* PLDS DVD+/-RW DS-8A8SH
* Integrated NIC 1 BRCM MBA Slot 0100 v15.2.10
* Slot 5 MLNX FlexBoot 3.4.159 (PCI 04:00.0)
* Slot 4 MLNX FlexBoot 3.4.162 (PCI 42:00.0)
Use Up/Down arrows to highlight desired item.
Use Enter to select highlighted item. Use Esc to continue normal boot.
Current User(s): root : 10.8.16.157

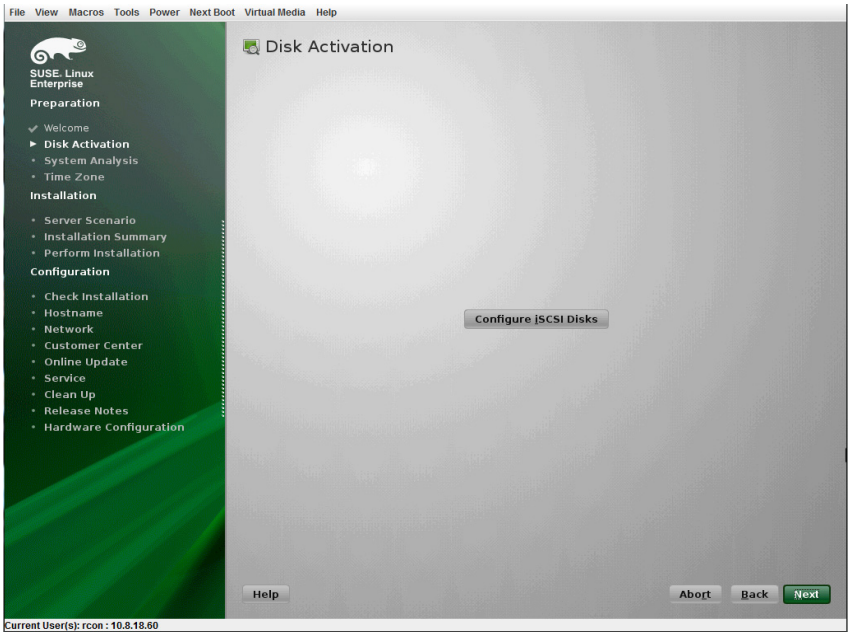
Step 3. Select the "Install SLES11.3" boot option from the menu (see pxelinux.cfg example above).

After about 30 seconds, the SLES installer will issue the notification below due to the PXELINUX boot label we used above.

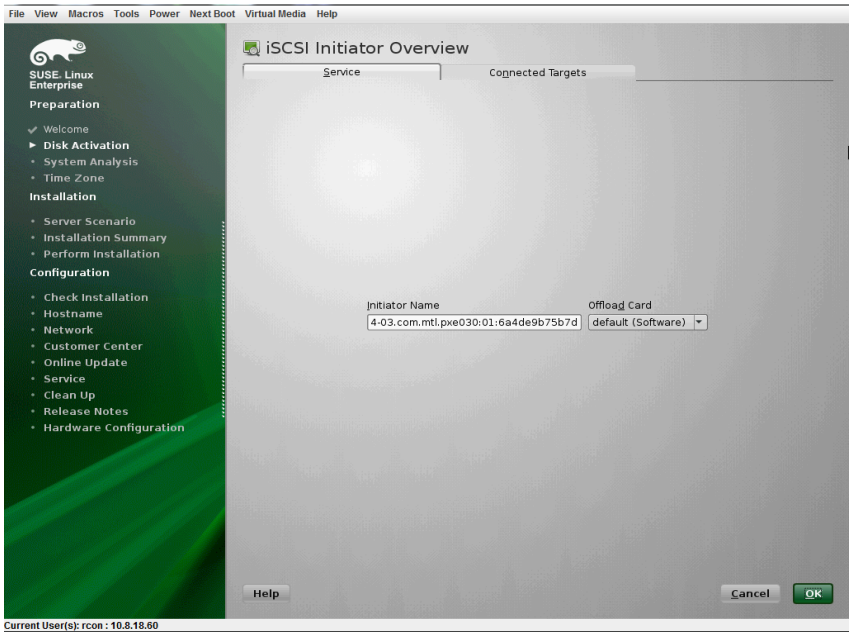

10.7.116.11, PowerEdge R720, User: root, 2.8 fps
File View Macros Tools Power Next Boot Virtual Media Help
>>> Linuxrc v3.3.91 (Kernel 3.0.76-0.11-default) <<<
Could not find the SUSE Linux Enterprise Server 11 Repository.
Activating manual setup program.
OK
Current User(s): root : 10.8.16.157

Step 4. Click OK.

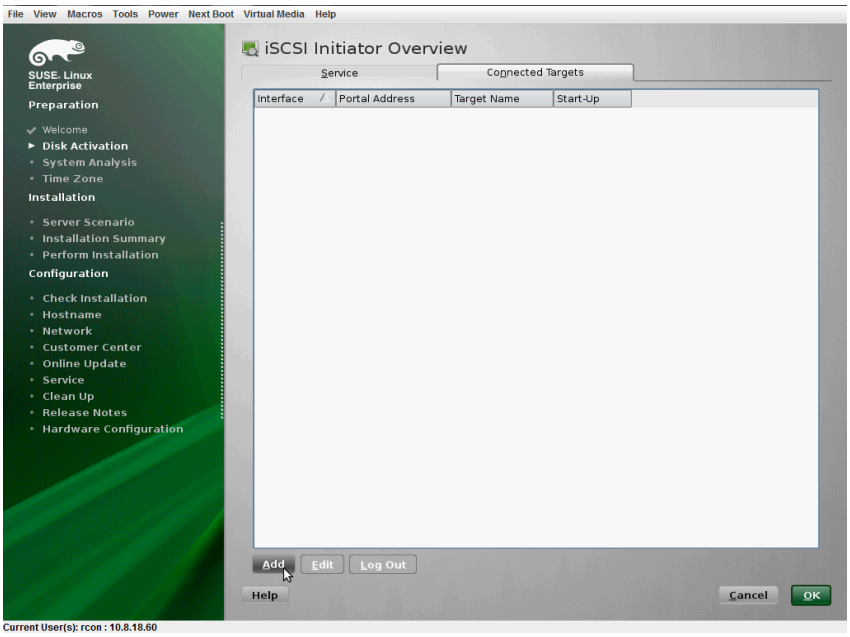
Step 5. Click on the **Configure iSCSI Disks** button.



Step 6. Choose **Connected Targets** tab.

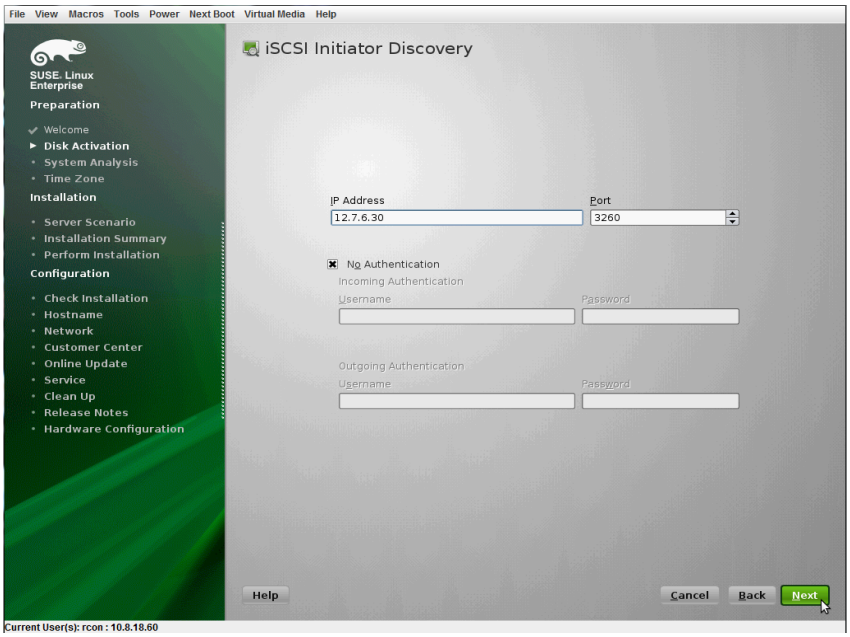


Step 7. Click **Add**.



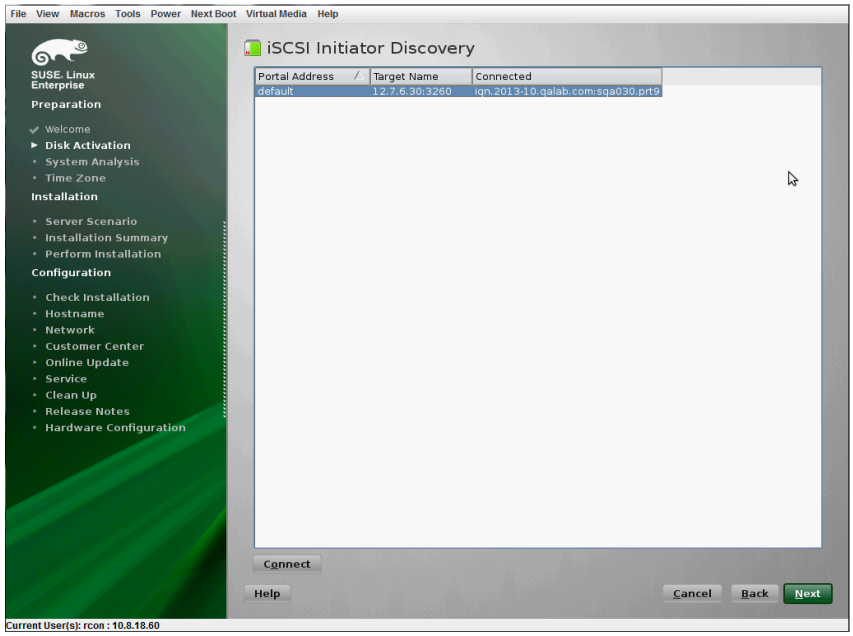
Step 8. Enter the IP address of the iSCSI storage target.

Step 9. Click **Next**.

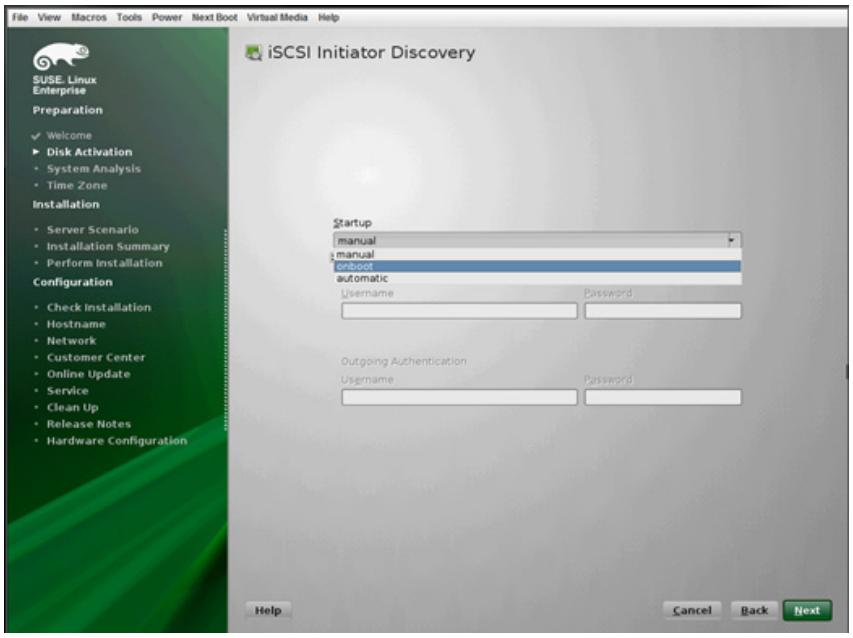


Step 10. Select the relevant target from the table (In our example, only one target exist so only one was discovered).

Step 11. Click **Connect**.

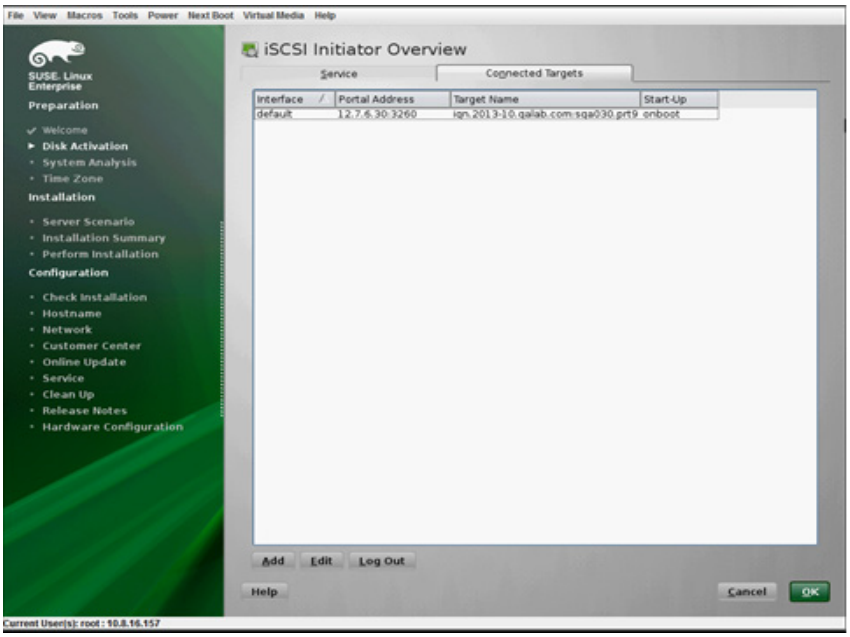


Step 12. Select **onboot** from drop-list.



Step 13. Click **Next** to exit the discovery screen.

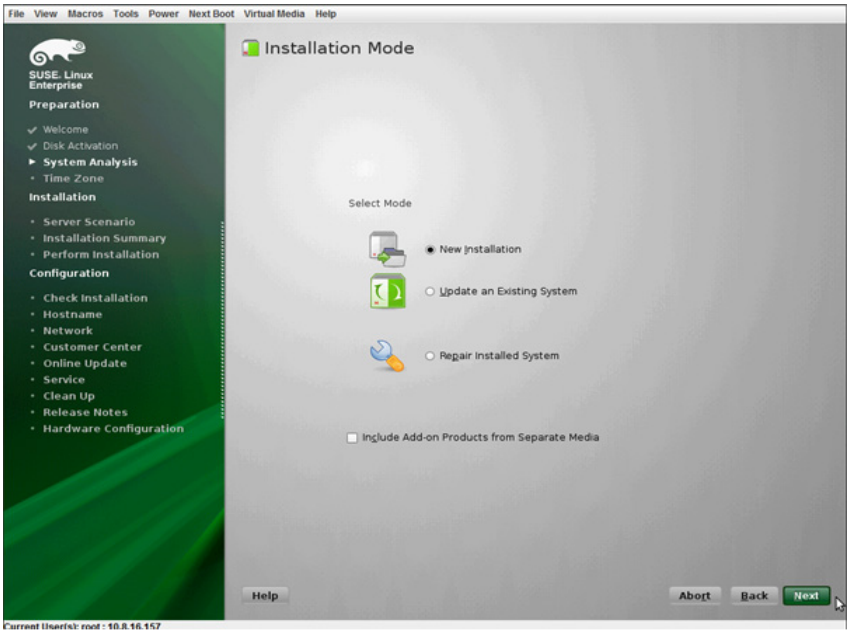
Step 14. Go to the **Connected Targets** tab again to confirm iSCSI connection with target.



Step 15. Click **OK**.

Step 16. Click **Next** back at the Disk Activation screen.

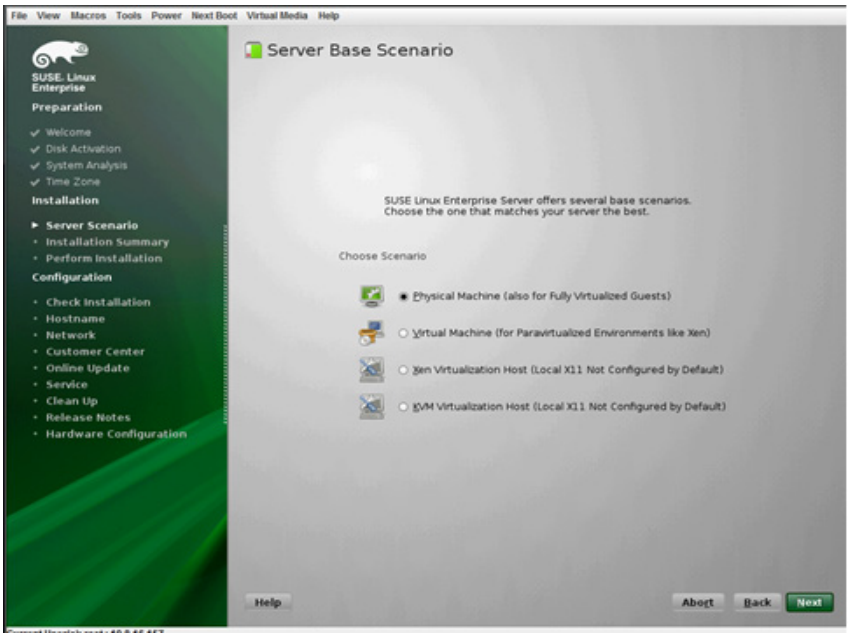
Step 17. Select **New Installation**.



Step 18. Click **Next**.

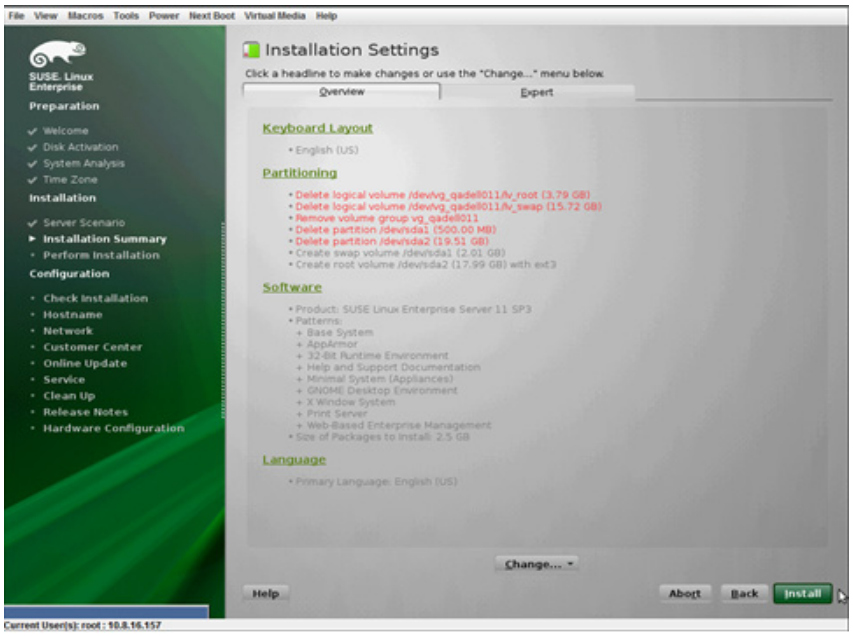
Step 19. Complete **Clock and Time Zone** configuration.

Step 20. Select **Physical Machine**.



Step 21. Click **Next**.

Step 22. Click **Install**.



Make sure `"open-iscsi"` RPM is selected for the installation under `"Software"`.

After the installation is completed, the system will reboot.

Make sure you choose `"SLES11.3x64_iscsi_boot"` label from the boot menu (See Section E.3, “Configuring the PXE Server”, on page 304).

Step 23. Complete post installation configuration steps.

It is recommended to download and install the latest version of MLNX_OFED_LINUX available from
http://www.mellanox.com/page/products_dyn?product_family=26&mtag=linux_sw_drivers

E.4.1 Using PXE Boot Services for Booting the SLES11 SP3 from the iSCSI Target

Once the installation is completed, the system will reboot. At this stage, it is expected from the client to perform another PXE network boot with FlexBoot®.

Make sure you choose the "SLES11.3x64 iSCSI boot" label from the boot menu (See Section E.3, “Configuring the PXE Server”, on page 304).

E.5 Installing RHEL6.4 on a Remote Storage over iSCSI

Step 1. Reboot the diskless client and perform a PXE boot with FlexBoot.

This is not an iSCSI boot, rather a regular PXE initiated network deployment of RHEL6.4. In the DHCP server configuration, the PXELINUX (pxelinux.0) and a RHEL 6.4 distribution media will be provided for network installation. For further information, please refer to sections Section E.6.2, “DHCP Configuration for PXELINUX with FlexBoot”, on page 318 and Section E.6.3, “pxelinux.cfg/default”, on page 319.

The clients’ HDD was removed beforehand; therefore the RHEL installer (A.K.A Anaconda) will ask to locate a HDD. The Anacaonda’s built-in iSCSI discovery will be used to connect to the iSCSI target LUN partition. For further information, please refer to: https://access.redhat.com/site/documentation/en-US/Red_Hat_Enterprise_Linux/6/html-single/Installation_Guide/index.html#ISCSI_disks

Step 2. Select the Network Interface (the same interface which was used by FlexBoot during PXE boot stage) for the installation process once prompted.

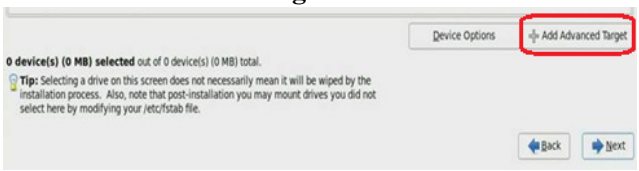
Step 3. Select the type of Installation Media access.

In this example, we use NFS, which also requires us to enter the NFS server name, and the directory path to the installation media on the NFS.

Step 4. Select **Specialized Storage Devices**.

What type of devices will your installation involve?
Basic Storage Devices
Installs or upgrades to typical types of storage devices. If you're not sure which option is right for you, this is probably it.
Specialized Storage Devices
Installs or upgrades to enterprise devices such as Storage Area Networks (SANs). This option will allow you to add FCoE / iSCSI / zFCP disks and to filter out devices the installer should ignore.

Step 5. Click on the + **Add Advanced Target** button.

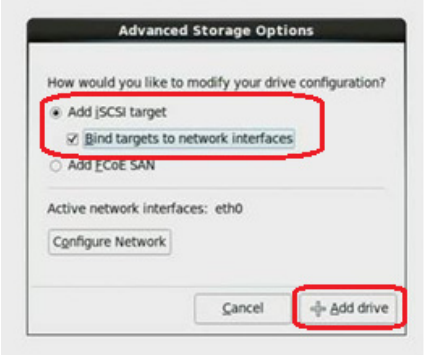

Device Options
Add Advanced Target
0 device(s) (0 MB) selected out of 0 device(s) (0 MB) total.
Tip: Selecting a drive on this screen does not necessarily mean it will be wiped by the installation process. Also, note that post-installation you may mount drives you did not select here by modifying your /etc/fstab file.
Back
Next

Step 6. In the Advanced Storage Options window perform the following:

Step a. Select the Add Iscsi Target option.

Step b. Check the **Bind targets to network interfaces** checkbox.

Step c. Click +**Add drive** button.



Step 7. Enter the IP address of iSCSI target.

Optionally, you may choose to enter a customized Initiator Name and select the necessary CHAP authentication of choice. Please refer to Section E.5.1, “SAN-Booting the Diskless Client with FlexBoot”, on page 316 for further information.

In the example below, iSCSI Initiator Name is left with the default value given by the installer and iSCSI discovery authentication is left with No authentication.

iSCSI Discovery Details
To use iSCSI disks, you must provide the address of your iSCSI target and the iSCSI initiator name you've configured for your host.
Target IP Address: 12.7.6.30
iSCSI Initiator Name:
What kind of iSCSI discovery authentication do you wish to perform:
No authentication
Cancel
Start Discovery

Step 8. Check the relevant Node Name to log in.

If as a result of the discovery, multiple Node Names are found, select the one that is relevant to you.

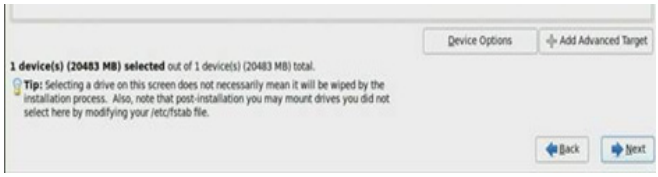
Step 9. Click **Login**.

A successful login is mandatory to proceed. A failure at this stage is probably a result of a target or network configuration error and recovery/troubleshooting that is out of the scope of this document.

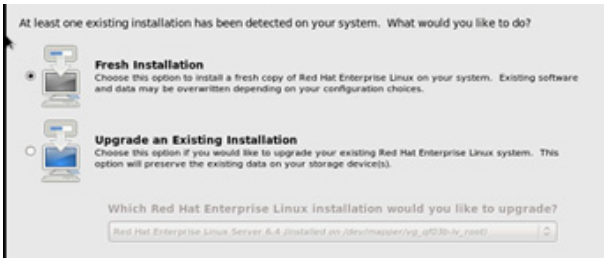
Step 10. Make sure a new storage LUN appears in the **Other AN Devices** tab.

A successful LUN discovery is mandatory to proceed. A failure at this stage is probably a result of a target or network configuration error and recovery/troubleshooting that is out of the scope of this document.

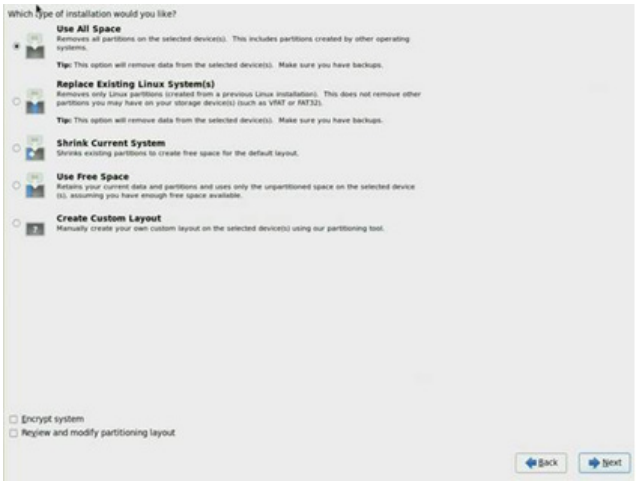
Step 11. Click **Next**.


Device Options
Add Advanced Target
1 device(s) (20483 MB) selected out of 1 device(s) (20483 MB) total.
Tip: Selecting a drive on this screen does not necessarily mean it will be wiped by the installation process. Also, note that post-installation you may mount drives you did not select here by modifying your /etc/fstab file.
Back
Next

Step 12. Select **Fresh Installation** and proceed with the Installation


At least one existing installation has been detected on your system. What would you like to do?
Fresh Installation
Choose this option to install a fresh copy of Red Hat Enterprise Linux on your system. Existing software and data may be overwritten depending on your configuration choices.
Upgrade an Existing Installation
Choose this option if you would like to upgrade your existing Red Hat Enterprise Linux system. This option will preserve the existing data on your storage device(s).
Which Red Hat Enterprise Linux installation would you like to upgrade?

Step 13. Select the **Use All Space** option.


Which type of installation would you like?
Use All Space
Removes all partitions on the selected device(s). This includes partitions created by other operating systems.
Tip: This option will remove data from the selected device(s). Make sure you have backups.
Replace Existing Linux System(s)
Removes only Linux partitions (created from a previous Linux installation). This does not remove other partitions you may have on your storage device(s) (such as VFAT or FAT32).
Tip: This option will remove data from the selected device(s). Make sure you have backups.
Shrink Current System
Shrinks existing partitions to create free space for the default layout.
Use Free Space
Retains your current data and partitions and uses only the unpartitioned space on the selected device (s), assuming you have enough free space available.
Create Custom Layout
Manually create your own custom layout on the selected device(s) using our partitioning tool.
Encrypt system
Review and modify partitioning layout
Back
Next

Step 14. Click **Next** and proceed with the Installation.

Step 15. Select the **Basic Server** option.

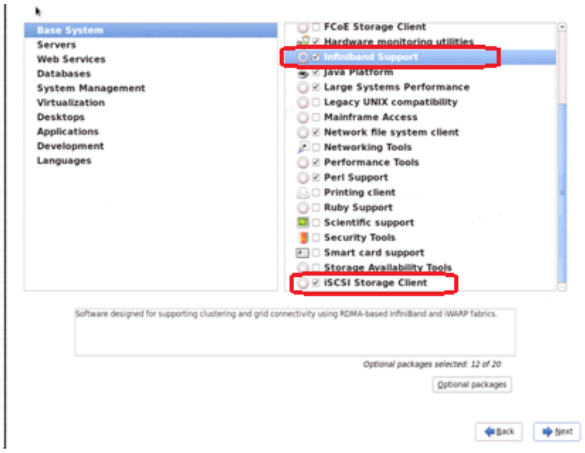
This is only one of the options that can be chosen, not the mandatory one.

The default installation of Red Hat Enterprise Linux is a basic server install. You can optionally select a different set of software now.
Basic Server
Database Server
Web Server
Identity Management Server
Virtualization Host
Desktop
Software Development Workstation
Minimal
Please select any additional repositories that you want to use for software installation.
Red Hat Enterprise Linux
Add additional software repositories
Modify repository
You can further customize the software selection now, or after install via the software management application.
Customize later
Customize now
Back
Next

Step 16. Check the **Customize Now** checkbox.

Step 17. Click **Next**.

Step 18. Select **Infiniband Support** and **iSCSI Storage Client**.



Step 19. Click **Next**. Allow the installation to reach completion.

E.5.1 SAN-Booting the Diskless Client with FlexBoot

When the installation process is completed, the client will ask to reboot. At that point, the DHCP server configuration for that client needs to be changed so that when it PXE boots again, it will get the root-path IQN and LUN information from the DHCP server.

For further information, please refer to section DHCP Configuration for iSCSI Boot with Flex-Boot (PXE SAN Boot).

Restart your DHCP service after changing the dhcp configuration file.

- Reboot the system

Congratulations, your Red Hat Enterprise Linux installation is complete.
Please reboot to use the installed system. Note that updates may be available to ensure the proper functioning of your system and installation of these updates is recommended after the reboot.
Back
Reboot

The expected result is that for the diskless PXE client to boot the newly installed RHEL6.4 from the iSCSI storage, and become an operational environment, accessible from any remote PC via ssh over 10GbE IP network

E.6 Sanity Checks

E.6.1 iSCSI Login

From a remote PC (called sqa070 below) with 10GE network connection to the iSCSI target, configure an iSCSI Initiator service and verify the correct target configuration by logging into the target. For CHAP configuration, refer to

Step 1. Install the initiator.

```
[root@sqa070 ~ ]# yum install -y iscsi-initiator-utils
```

Step 2. Configure the Initiator.

```
[[root@sqa070 ~ ]#  vim /etc/iscsi/iscsid.conf
node.startup = automatic
## Optional: for CHAP authentication, uncomment the following #lines
#discovery.sendtargets.auth.authmethod = CHAP
#discovery.sendtargets.auth.username = joe
#discovery.sendtargets.auth.password = secret

#node.session.auth.authmethod = CHAP
#node.session.auth.username = jack
#node.session.auth.password = 12charsecret


node.session.timeo.replacement_timeout = 120
node.conn[0].timeo.login_timeout = 15
node.conn[0].timeo.logout_timeout = 15
node.conn[0].timeo.noop_out_interval = 10
node.conn[0].timeo.noop_out_timeout = 15
node.session.initial_login_retry_max = 10
node.session.cmds_max = 128
node.session.queue_depth = 32
node.session.iscsi.InitialR2T = No
node.session.iscsi.ImmediateData = Yes
node.session.iscsi.FirstBurstLength = 262144
node.session.iscsi.MaxBurstLength = 16776192
node.conn[0].iscsi.MaxRecvDataSegmentLength = 131072
discovery.sendtargets.iscsi.MaxRecvDataSegmentLength = 32768
node.session.iscsi.FastAbort = No
```

Step 3. Start the iSCSI Initiator service.

```
[root@sqa070 ~ ]#  service iscsi start
```

Step 4. Discover the iSCSI target host.

In the example below, the IP address 12.7.6.30 is the iSCSI target.

```
[root@sqa070 ~ ]# iscsid start
[root@sqa070 ~ ]# iscsiadm -m discovery -t st -p 12.7.6.30
Starting iscsid:                                           [  OK  ]
12.7.6.30:3260,1 iqn.2013-10.qalab.com:sqa030.prt9
```

Achieving a successful target discovery at this stage is mandatory for proceeding with the process of iSCSI boot. A failure at this stage is probably a result of an erroneous target or network configuration and troubleshooting that is out of the scope of this document.

Step 5. Log into the target.

```
[root@sqa070 ~ ]# iscsiadm -m node -p 12.7.6.30  -T iqn.2013-10.qalab.com:sqa030.prt9
--login
Logging in to [iface: default, target: iqn.2013-10.qalab.com:sqa030.prt9, portal:
12.7.6.30,3260] (multiple)
Login to [iface: default, target: iqn.2013-10.qalab.com:sqa030.prt9, portal:
12.7.6.30,3260] successful.
```

A successful LUN login at this stage is mandatory for proceeding with the process of iSCSI boot. A failure at this stage is probably a result of an erroneous target or network configuration and troubleshooting that is out of the scope of this document.

Step 6. Verify the remote partition appears to the initiator as a local HDD.

```
[root@sqa070 ~ ]# fdisk -l

Disk /dev/sda: 500.1 GB, 500107862016 bytes
255 heads, 63 sectors/track, 60801 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x000518f2

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1   *           1         131     1048576   83  Linux
Partition 1 does not end on cylinder boundary.
/dev/sda2             131        2681    20480000   83  Linux
/dev/sda3            2681        2936     2048000   82  Linux swap / Solaris

Disk /dev/sdb: 21.5 GB, 21478670336 bytes
64 heads, 32 sectors/track, 20483 cylinders
Units = cylinders of 2048 * 512 = 1048576 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00000000
```

E.6.2 DHCP Configuration for PXELINUX with FlexBoot

The following DHCP configuration is presented as is, and may not work in all environments.

```
authoritative;
ddns-update-style none ;
ddns-updates off ;
allow bootp;
always-broadcast off ;
always-reply-rfc1048 off;
boot-unknown-clients on ;

option client-system-architecture             code 93 = unsigned integer 16;
option vendor-encapsulated-options            code 43 = string;
option vendor-class-identifier                code 60 = string;


class "PXEClient" {
                match if substring (option vendor-class-identifier, 0, 9) = "PXECli-
ent";
                option vendor-class-identifier "PXEClient";
option vendor-encapsulated-options 06:01:08 ;
                option dhcp-parameter-request-list = concat(option dhcp-parameter-
request-list,43);

}
```

```
subnet 12.7.0.0 netmask 255.255.0.0 {
    option dhcp-server-identifier 12.7.6.30 ;
                option domain-name "pxe030.mtl.com" ;
                option domain-name-servers 12.7.6.30 ;
                default-lease-time 86400 ; # 1 day
                max-lease-time 86400 ;
                option ntp-servers 12.7.6.30;
}



host sqa070 {
                fixed-address 12.7.6.70   ;
                hardware ethernet 00:02:c9:32:e8:80 ;
                next-server 12.7.6.30;
if option client-system-architecture = 00:00 {  filename "pxelinux.0" ; }

}
```

E.6.3 pxelinux.cfg/default

```
[root@sqa030 ~]# cat /var/lib/tftpboot/pxelinux.cfg/default
LABEL rh6.4x64_instl_manual
MENU LABEL Manual Installation RHEL6.4
                KERNEL RHEL6.4-x86_64-DVD1/images/pxeboot/vmlinuz
                APPEND initrd=RHEL6.4-x86_64-DVD1/images/pxeboot/initrd.img
```

E.6.4 DHCP Configuration for iSCSI Boot with FlexBoot (PXE SAN Boot)

Modify the following host declaration in your DHCP configuration file to allow PXE SAN Boot

```
host sqa070 {
filename "";
     option root-path "iscsi:12.7.6.30::::iqn.2013-10.qalab.com:sqa030.prt9";
     fixed-address 12.7.60.70  ;
     hardware ethernet 00:02:c9:32:e8:80  ;
}
```

E.7 SAN-Booting with FlexBoot in CHAP Environment

For Successful SAN-Booting with FlexBoot in CHAP environment, the FlexBoot in use must support passing CHAP credentials. Please note that FlexBoot v3.4.149 does not support such operation. Hence, use FlexBoot to chain-load an UNDI software which supports passing CHAP credentials. This can be achieved by using the UNDI software module from iPXE.org, called undionly.kpxe. A Linux host is required to create undionly.kpxe.

Step 1. Install the below prerequisite software to support necessary UNDI's compilation.

```
[root@sqa030 ~]#  yum install -y gcc binutils make perl
```

Step 2. Download the UNDI sources from iPXE.org.

For more information, visit http://www.ipxe.org/download

```
[root@sqa030 ~]#  git clone git://git.ipxe.org/ipxe.git
```

Step 3. Edit a command file named sanbootnchap.ipxe (the name is given as an example whereas the .ipxe file extension is mandatory) with the following lines.

Make sure to enter your own values for username and password per your CHAP configuration. For reasons of simplicity, and coherence with this document examples, we gave our CHAP the username joe, and the password secret. Note that these CHAP settings are going to be hardcoded into this module and any modification done to them will require you to recompile the module after changing sanbootnchap.ipxe.

```
[root@sqa030 ~]#  cd ipxe/src
[root@sqa030 src]# vim sanbootnchap.ipxe
#!ipxe
dhcp || dhcp || exit
set username joe
set password secret
echo ${root-path}
isset ${root-path} || exit
echo Booting from iSCSI tgt
sanboot --no-describe  ${root-path}
```

Step 4. Compile the undionly.kpxe module.

For more information, refer to http://www.ipxe.org/howto/chainloading

```
[root@sqa030 src]# make bin/undionly.kpxe EMBED=sanbootnchap.ipxe
```

Step 5. Copy the bin/undionly.kpxe to your TFTP root directory (E.G.: /var/lib/tftpboot).

Step 6. Edit the client's host declaration in the DHCP configuration file for chain-loading undionly.kpxe.

The outcome of this procedure is to have FlexBoot download undionly.kpxe to the client's RAM, and then have undionly.kpxe authenticate iSCSI and login with the iSCSI target.

```
host sqa070 {
                  next-server 12.7.6.30;
if option client-system-architecture = 00:00 {  filename "undionly.kpxe" ; }
fixed-address 12.7.60.70  ;
hardware ethernet 00:02:c9:32:e8:80  ;
if exists user-class and option user-class = "iPXE" {
                                  option root-path "iscsi:12.7.6.30::::iqn.2013-
10.qalab.com:sqa030.prt9";
                                  filename "";
                  }
}
```

For CHAP users: All the CHAP authentication lines mentioned as comments in the iSCSI target and initiator configuration examples in sections Section E.1, “Configuring the iSCSI Target Machine”, on page 302 and Section E.6.1, “iSCSI Login”, on page 316 must be un-commented beforehand, and during RHEL installation, follow these steps:

iSCSI Discovery Details
To use iSCSI disks, you must provide the address of your iSCSI target and the iSCSI initiator name you've configured for your host.
Target IP Address: 12.7.6.30
iSCSI Initiator Name: iqn.1994-05.com.domain:01.e26b2e
What kind of iSCSI discovery authentication do you wish to perform:
CHAP pair
CHAP Username: joe
CHAP Password: ●●●●●●
Cancel
Start Discovery